IP Protection and Supply Chain Security through Logic Obfuscation: A Systematic Overview

KAVEH SHAMSI, ECE Department, University of Florida
MENG LI, ECE Department, University of Texas at Austin
KENNETH PLAKS, Defense Advanced Research Projects Agency
SAVERIO FAZZARI, Booz Allen Hamilton
DAVID Z. PAN, ECE Department, University of Texas at Austin
YIER JIN, ECE Department, University of Florida

The globalization of the semiconductor supply chain introduces ever-increasing security and privacy risks. Two major concerns are IP theft through reverse engineering and malicious modification of the design. The latter concern in part relies on successful reverse engineering of the design as well. IC camouflaging and logic locking are two of the techniques under research that can thwart reverse engineering by end-users or foundries. However, developing low overhead locking/camouflaging schemes that can resist the ever-evolving state-of-the-art attacks has been a challenge for several years. This article provides a comprehensive review of the state of the art with respect to locking/camouflaging techniques. We start by defining a systematic threat model for these techniques and discuss how various real-world scenarios relate to each threat model. We then discuss the evolution of generic algorithmic attacks under each threat model eventually leading to the strongest existing attacks. The article then systematizes defences and along the way discusses attacks that are more specific to certain kinds of locking/camouflaging. The article then concludes by discussing open problems and future directions.

CCS Concepts: • **Security and privacy → Hardware attacks and countermeasures**; *Hardware security implementation*;

Additional Key Words and Phrases: Hardware security, logic obfuscation, logic locking, IC camouflaging

ACM Reference format:
Kaveh Shamsi, Meng Li, Kenneth Plaks, Saverio Fazzari, David Z. Pan, and Yier Jin. 2019. IP Protection and Supply Chain Security through Logic Obfuscation: A Systematic Overview. *ACM Trans. Des. Autom. Electron. Syst.* 24, 6, Article 65 (September 2019), 36 pages.
https://doi.org/10.1145/3342099

1 INTRODUCTION

In recent decades, we have witnessed the globalization of the integrated circuit (IC) supply chain propelled by the ever-increasing design complexity and cost of the semiconductor industry. However, such globalization comes at a cost. Although it reduces the overall expense, the worldwide distribution of IC design, fabrication, assembly, and deployment introduces serious IP privacy and

Authors' addresses: K. Shamsi and Y. Jin (corresponding author), ECE Department, University of Florida, Gainesville, Florida; emails: kshamsi@ufl.edu, yier.jin@ece.ufl.edu; M. Li and D. Z. Pan, ECE Department, University of Texas at Austin, Austin, Texas; emails: meng_li@utexas.edu, dpan@ece.utexas.edu; K. Plaks, Defense Advanced Research Projects Agency, Arlington, Virginia; email: kenneth.plaks@darpa.mil; S. Fazzari, Booz Allen Hamilton, Arlington, Virginia; email: saverio.fazzari.ctr@darpa.mil.

1084-4309/2019/09-ART65 $15.00
https://doi.org/10.1145/3342099

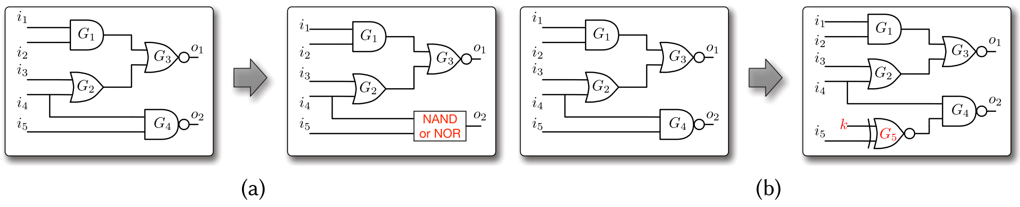


Fig. 1. Example of (a) IC camouflaging and (b) logic locking.

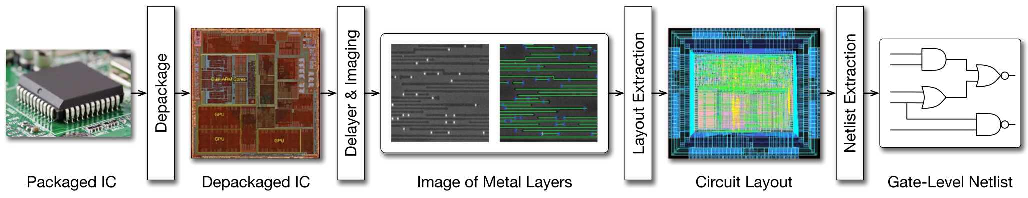


Fig. 2. Physical reverse engineering flow: An adversary can depackage and delayer an IC and leverage image processing techniques to reconstruct the circuit layout. Then the gate-level netlist can be extracted.

integrity infringement. The primary risks associated with the globalized IC supply chain can be broadly categorized as (1) malicious design modification [34, 44] and (2) IP theft through reverse engineering [45, 58, 88, 89, 91].

The threat of reverse engineering and IP theft arises from the fact that the design is disclosed to potentially malicious adversaries, including untrusted foundries and end-users. As for the foundries, they have full knowledge of the circuit layout and hence can extract the transistor-level design directly [58, 88, 91]. As for malicious end-users, per Figure 2, given a packaged IC acquired from the market, after depackaging, delayering, imaging, and processing these images, the circuit layout can be reconstructed [91]. The process of layout reconstruction and netlist extraction is usually referred to as physical reverse engineering. Over the past decade, such reverse engineering techniques have developed rapidly, demonstrating successful reconstruction of products of leading semiconductor companies in advanced technology nodes [16]. Therefore, to protect hardware IP, one must protect the design against reverse engineering attacks. In fact, the more serious threat of malicious modification itself relies on a certain level of successful reverse engineering of the design in the first place.

There are three main broad categories of techniques for thwarting reverse engineering by end-users or foundries: (1) Split-manufacturing, (2) Logic locking, and (3) IC camouflaging. Split-manufacturing is based on fabricating the upper metal layers of the IC in a trusted foundry. Locking and camouflaging are two techniques that try to thwart reverse engineering in the absence of a trusted nano-fabrication facility and are the focus of this article. Both these techniques can be modeled mathematically as disclosing a partially ambiguous design to attackers. Hence in this article, we sometimes refer to both of them as "logic obfuscation," even though they have important practical differences that must be noted [19, 43, 60, 69, 76, 105, 114].

IC camouflaging is based on using fabrication-level techniques to build circuits whose functionality cannot be easily deduced using known physical reverse engineering techniques [19, 43, 60, 105]. This typically means creating layout structures that cannot be deduced to a specific functionality from the top view under nanoscale microscopy. For digital circuits, this can be done by first designing smaller camouflaged units/cells and then inserting them throughout the netlist with a given insertion strategy. As shown in Figure 1(a), a camouflaging cell that from the top view appears to implement either a NAND or a NOR function, is inserted into the netlist to replace the original gate G_4. Since the functionality of the camouflaging cell cannot be determined in the reverse engineering process, the netlist's functionality is obscured.

Logic locking is based on adding some form of programmability to the design and ensuring that the circuit cannot function properly without the circuit being programmed with a secret string of configuration data referred to as the "key" [69, 95, 99, 100, 103]. Figure 1(b) shows an example of logic locking in which an additional "key-input" along with a "key-gate" is added. The correct circuit functionality only manifests itself upon programming of the correct key bit ($k_1 = 1$).

In the past few years, research on logic obfuscation has received considerable attention and has made remarkable progress. This research can be categorized into different hierarchical levels: fabrication level [9, 17, 18, 93], cell level [27, 43, 49, 61], and netlist level [39, 43, 61, 77, 78, 105]. At the fabrication level, the main focus is on the nano-device structures that can hide the circuit's functionality. For camouflaging, this includes using doping levels, dummy-contacts between metal layers, and so on. For locking at the device level, tamper-resistant programmable technologies are needed. At the cell level, the nano-primitives are used to develop different cell designs that can manifest reverse engineering ambiguity. The next step is to transform the original circuit and incorporate these cells with the goal of maximizing security with minimum area/delay/power overhead. The majority of the algorithmic work involved with locking/camouflaging hence focuses on a netlist-level abstraction [69, 99, 100, 103]. With a netlist-level abstraction both types of obfuscation (camouflaging and locking) can be modeled with the same mathematics resulting in attacks that can target both techniques interchangeably.

The main distinction between IC camouflaging and logic locking is in that since IC camouflaging requires fixed ambiguous layout structures, these details must be disclosed to the foundry for the purpose of fabricating them. Hence, IC camouflaging can only protect the hardware IP against malicious end-users [61]. However, in logic locking, a post-fabrication activation step is required to program the secret key into the circuit [69] without which the foundry should not be able to fully operate the circuit [69].

Locking and camouflaging schemes can be attacked with different approaches. If the nano-device itself cannot be reverse engineered, then algorithmic attacks can be used at the netlist level, typically referred to as "deobfuscation" algorithms/attacks. Different deobfuscation algorithms have been proposed under different threat models [26, 61, 75, 81, 83, 84, 87, 96, 97, 106, 111, 116]. In response to the fast-evolving array of deobfuscation attacks, new obfuscation schemes are developed accordingly. This "arms race" results in stronger attacks that inspire stronger obfuscation schemes that, in turn, inspire new attacks.

Given this context, it is timely to conduct a comprehensive review of recent progress in the field of logic obfuscation. To the best of our knowledge, there is no dedicated comprehensive review on logic obfuscation that considers the most recent radical changes in the field within a systematic threat model. As we will discuss in the article, the lack of a clear systematic approach to threat modeling has resulted in various schemes that claim qualitative security but are in fact not secure once a more accurate threat model is considered. Having a taxonomy of threat models helps designers understand under what conditions exactly the locking/camouflaging scheme guarantees what level of security. Therefore, this survey article starts with a deep discussion about what logic obfuscation aims to protect and an enumeration of threat models that relate to real-world scenarios in different ways. It then moves on to develop a map of deobfuscation attacks with the goal of describing a collection of known attacks that form the frontier of deobfuscation within each respective threat model. The article then does the same with defense schemes, finally arriving at a map that allows the reader to get a sense of what one can expect from the state of the art of locking and that of camouflaging, were they to transfer the existing research to practice. The survey concludes by discussing some common pitfalls in the area and open research challenges.

The article is organized as follows. Section 2 describes the attack model of logic obfuscation and provides an overview of the evolution of attack and defense schemes. Section 3 surveys attack algorithms. Section 4 describes protection schemes. Section 5 summarizes the state of the art and presents pitfalls and future directions. Section 6 concludes the article.

2 AN OVERVIEW OF OBFUSCATION RESEARCH

Over the past 10 years, extensive research has been conducted for both logic obfuscation and deobfuscation attacks. Before we explain these in more detail, we first provide a dense overview of this arms-race. The first significant logic locking strategy, EPIC, is proposed in Reference [69]. EPIC randomly inserts key-controlled XOR/XNOR gates into the netlist and proposes a key distribution framework using public-key cryptography. The original EPIC paper claimed security by arguing that the Quantified Boolean Formula (QBF) that results from trying to find a key that equates the obfuscated circuit to the original circuit was intractable due to alternating quantifiers for larger key sizes. Following EPIC, Reference [7] proposed a new locking strategy that leverages key-controlled reconfigurable logic blocks, denoted as barriers. Compared with EPIC, Reference [7] further discussed the need to ensure that all paths in the circuit from inputs to go through at least some key logic to heuristically increase security.

The first sophisticated deobfuscation scheme called the testing-based attack was proposed in Reference [61] and was further enhanced in Reference [39]. This for the first time opened up the discussion on threat modeling. The approach assumes the attacker has access to all internal registers of a functional/unlocked circuit and can query its combinational slices with chosen input patterns and observe the correct output. This threat model was later called the "oracle-guided" model. Instead of exhaustively searching the entire input space, the testing-based attack leverages hardware testing principles, e.g., sensitization and justification, to select input vectors to query the functional IC. This approach has been effective at attacking circuits obfuscated by random-insertion by isolating the key bits. Later, it was observed that when different key bits interfere with each other, as described in detail in Section 4, the testing-based attack can fail. Therefore, Reference[61] proposes a clique-based obfuscation strategy to insert key-gates that form a clique in the dependency graph of the circuit so that the number of interfering key pairs are maximized.

The Boolean satisfiability solver (SAT)-based attack was proposed in Reference [26, 87, 111] as a general oracle-guided attack. In the SAT attack, the problem of finding what patterns to query and resolving the key based on the queries, is formulated into a series of incremental SAT problems. By solving the SAT problem, new input patterns called discriminating input patterns (DIPs) are mined and queried on the oracle. The resulting input/output (I/O)-pairs are then added back into the SAT problem, allowing the algorithm to implicitly prune incorrect keys. The I/O-pair addition guides the DIP mining process until a correct key is found. Exploiting the performance of modern SAT solvers the SAT attack was able to deobfuscate the majority of existing low-overhead locking and camouflaging schemes at the time on benchmark circuits with thousands of gates.

Shortly after, a new approach to obfuscation was proposed to thwart the SAT-attack based on decreasing the effectiveness of each DIP, in essence making each DIP prune fewer keys and thus increasing the minimum number of DIPs required to find a correct key [43, 99, 104, 105, 108]. These approaches relied heavily on the use of "point-functions" or comparator logic. By inserting these blocks in the circuit they cause the obfuscated circuit to misbehave on only a small number of input patterns. Since reverse engineering the circuit requires finding these few misbehaving locations in the exponentially large input space and patching the output for them, it can be shown that any oracle-guided deobfuscation methods will require on average exponential queries for perfectly deobfuscating such a scheme.

As may be noticed from the above description, the main challenge with point-function schemes is that they do not create a high output error. That is, the obfuscated circuit deviates from the original circuit on only an exponentially small portion of the input space. In fact the authors of point-function schemes were familiar with this flaw and hence proposed combining traditional non-SAT-resilient schemes that had higher error-rates with the SAT-resilient point-function schemes. However, later, other SAT attack variants such as AppSAT [75, 80] and DDIP [84] where proposed that were capable of taking such compound obfuscation schemes and attacking the high-error portion in minutes reducing the high-error SAT-resilient compound scheme back to a low-error point-function scheme. AppSAT specifically was called an approximate attack as it focused on the exponential approximation of the oracle rather than perfect deobfuscation. This view of obfuscation opened up studying a new notion of security called approximation-resiliency. To respond to approximate attacks, researchers have tried to create schemes that consider both the output error probability as well as the minimum DIP count.

Another approach to securing obfuscation has been to introduce unconventional structures into the obfuscated circuit [77, 78, 114]. In References [65, 66, 77, 78], circuit wire interconnections are obfuscated by introducing dense, nested, cyclic structures into the netlist, which do not impact the circuit functionality but significantly complicate the attack process. In another effort [114], some flip-flops in the circuit are deliberately removed to create a mixture of unconventional timing paths in the circuit, which cannot be directly resolved by baseline SAT attacks.

Cyclic obfuscated circuits were first broken with the CycSAT algorithm proposed in References [116]. It was later shown first in Reference [64] and then in References [67, 82] that CycSAT attacks may run into problems when attacking dense cyclic circuits and exhaustive cycle enumeration of such circuit may be necessary, which is an exponential computational task. As for timing-based camouflaging schemes, TimingSAT [42] was proposed and tested on benchmark circuit as well.

As for logic obfuscation in the realm of sequential logic, it was first noted in References [25, 53] that in the absence of access to all of the oracle's state-elements a sequential formulation of the SAT attack is required. This formulation is largely similar to the oracle-guided SAT attack, except that SAT queries are replaced with model-checking queries and unrolling the sequential circuit over several clock cycles. The model-checking attack was shown to be successful in deobfuscating large benchmarks although with significant slow-down compared to the combinational SAT attack at least up to tens of cycles.

In parallel with all the above work on combinational obfuscation an entirely separate branch of obfuscation focused on transforming and obfuscating finite-state-machines (FSMs) [3, 13, 14, 54, 112]. By adding additional dummy states to an FSM and then having the user traverse the dummy states to reach the original FSM, a locking mechanism can be implemented. This branch of obfuscation did not receive much attention from attackers, hence a clear in-depth threat model and analysis was never developed until recently, where it was shown that the majority of these FSM obfuscation schemes may not be able to resist even attacks that do not have access to an oracle circuit known as oracle-less attacks. A couple of oracle-less attacks [12, 51] have been somewhat successful against older combinational locking schemes as well.

The description above provides a high-level timeline of the major milestones in obfuscation research. In the following sections, rather than the above time-view we will devise a categorization based on the attackers' capabilities and the defenders goals. We will discuss the frontier in each area in more detail.

3 ON ATTACKS

3.1 Notations

Before we begin our discussion on attacks, we review some basic mathematical notation.

We let c_o denote the original circuit before obfuscation. As for combinational circuits, c_o can be modeled as a Boolean function/circuit that has n input bits with the input space $\mathcal{I} = \{0,1\}^n$ and m output bits with output space $\mathcal{O} \subseteq \{0,1\}^m$. Logic locking can be modeled as transforming c_o to an augmented function $c_e : \mathcal{I} \times \mathcal{K} \rightarrow \mathcal{O}$ by adding l key inputs, where $\mathcal{K} \subseteq \{0,1\}^l$ denotes the key space. Meanwhile, there always exists at least one correct key vector $k^* \in \mathcal{K}$ such that $c_e(i, k^*) = c_o(i), \forall i \in \mathcal{I}$. The additional variables k induce a function space $C = \{c_e(i,k) | k \in \mathcal{K}\}$.

While IC camouflaging has a different implementation, it can be modeled with the same semantics. We can encode the ambiguity in the camouflaged netlist using key-variables. For instance, a camouflaged cell that implements either AND/NAND can be modeled with an AND gate plus a NAND gate that is arbitrated with a key-controlled MUX gate. The key-variable decides whether to pick the AND or NAND gate. It can be shown that all camouflaging schemes can be encoded with a polynomial number of key-variables [26, 110]. Therefore, it is possible to study attacks on camouflaging and locking within the same context.

If the original or obfuscated circuit are sequential, then c_o and c_e become stateful Boolean functions. A sequential circuit takes input i, produces output o, and updates an internal state s in each clock cycle. Such a circuit can be expressed by a next-state function $ns(i,s) \rightarrow s'$ and an output function $of(i,s) \rightarrow o$. Hence, we can write $c_e(i,k) : (s', o) \leftarrow (ns_e(i,k), of_e(i,s,k))$, or simply write $c_e(i,s,k)$ to denote that c_e is a sequential function with primary input i, key input k, and current state s that produces an output and updates s.

A widely used notion in obfuscation is output error rate. Error rate can be defined in various ways. One way is to take the probability of the obfuscated circuit disagreeing with the original circuit over the combined input and key space ($\mathcal{I} \times \mathcal{K}$):

$$\mathrm{Er}(\mathrm{c_e}, \mathrm{c_o}) = \left| \Pr_{i \in I, k \in K} \left[c_e^m(i,k) \neq c_o^m(i) \right] - \frac{1}{2} \right|$$

Note that this probability can be defined over inputs on a fixed key or keys on a fixed input as well. These various definitions capture similar notions for most obfuscation schemes.

As for FSM obfuscation the state-transition-graph (STG) of an FSM can be modeled with a graph with a state-encoding on each node and a Boolean condition on the input on each edge. The FSM boots up in the reset state and each clock cycles makes transitions based on the edge conditions. This graph will be transformed to another graph as a result of obfuscation.

3.2 Modeling Attackers

The threat/attack model for a given security problem defines clearly the capabilities and intentions of the attacker. The threat model in logic obfuscation is extremely critical to the security analysis of a particular scheme. While in some threat models it is easy to secure logic obfuscation with very low overhead, in other threat models it might be impossible to achieve security at all. Hence it should be the first step of anyone who is employing or studying an obfuscation scheme to accurately specify the intended threat model.

First, with respect to having access to an oracle, the following categories can be enumerated:

- **Oracle-less (OL)**: The attacker has access only to the obfuscated netlist c_e [51]. For a foundry attacker this would come from converting the GDS layout files to a transistor-level and then logic-level netlist. For an end-user, c_e must be extracted from physical reverse engineering. This can be seen in Figure 3.
- **Oracle-guided (OG)**: In addition to the obfuscated netlist, the attacker has access to a functional circuit, or "oracle," that implements the original circuit c_o [26, 87]. This can be acquired from the open market and is treated as a black-box circuit. The attacker cannot

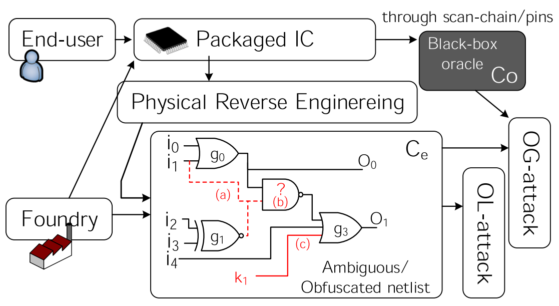


Fig. 3. The foundry has access to the layout of the IC and hence all non-programmable parts of the netlist are visible to the foundry. The end-user will have to physically reverse engineer a packaged IC to extract a netlist that will have ambiguities around camouflaged logic. Given access to the ambiguous netlist c_e, the oracle-guided (OG) and the oracle-less (OL) attackers want to recover the original circuit c_o.

observe or probe the internal signals of this chip directly but has access to all state-elements in the functional circuit and can hence query the original circuit's combinational blocks/cones with chosen input patterns and observe their output. In this attacker model there can be no state-elements in between a pair of controllable-observable points.
- **Sequential Oracle-guided (SOG)**: In this case, the attacker has access to an oracle but cannot control/observe some internal state elements [25, 52]. In this case, the logic between a controllable-observable pair of points is sequential. Hence the attacker has input-output access to a sequential c_o. The reset state of this chip can also be unknown.
- ***t*-probed Oracle**: When the attacker is allowed to probe $t \geq 1$ locations in the oracle circuit. It is easy to see that secure obfuscation under this model should be impossible with low overhead as the attacker can probe the camouflaged-cells/keys or their fanouts. If the entire computation is randomized, then the source of the randomization can still be probed to reverse engineer the circuit.

Each of the above threat models map to a real-world scenario. For instance, in IC camouflaging a non-oracle-guided threat model may not be reasonable for commercial ICs. Since commercial ICs are manufactured in batch, if an attacker has access to one chip that can be reverse engineered to extract the obfuscated netlist, it is reasonable to assume that he/she can get access to a second chip from the same market to query as an oracle. As for logic locking, a foundry that is fabricating the first-ever batch of a locked circuit does not have access to an oracle as no oracle has yet been programmed and marketed. However, an end-user that is trying to reverse engineer a commercial locked circuit is likely to have access to a second chip as an oracle. Opto-electrical probing from the backside [47, 90] or top-layer physical probing are increasingly being successful in reading active voltage levels on deeply scaled technologies. If there is no physical countermeasures against such techniques (secure coating, etc. [68, 92]), then a t-probed-oracle model is necessary, which implies extremely difficult if not impossible protection. As another important example, the leakage of testing data (correct input/output patterns) can be considered as a constrained oracle-guided model [107].

We can also enumerate attackers in terms of their ability in recognizing and reverse engineering ambiguous elements in a locked/camouflaged IC:

- **Distinct Ambiguity**: The locked circuit attacker knows which wires in the locked circuit are keys but does not know the value of the keys. The camouflaged circuit attacker knows the cells and vias in the camouflaged circuit that are camouflaged/vague but does not know their exact functionality.

- **Ubiquitous Ambiguity**: The locked circuit attacker does not know which wires are programmable keys and which wires are normal inputs. The camouflaged circuit attacker does not know which vias/cells are camouflaged. As a result, the attacker suspects that all elements may be fake and all inputs may be keys.

In logic locking, for instance, if metal-to-metal antifuse is used for locking, then it is difficult to hide the location of keys if the antifuse layout is different than that of a normal via under microscopy. If an on-chip nonvolatile memory is used for programming keys, then we can try to make it such that the nonvolatile memory is used for other purposes in the circuit as well, so that an attacker cannot isolate the memory unit to find the key inputs. As for IC camouflaging, by creating cells and vias that look similar to normal cells and vias from the top view we can create a scenario where the attacker suspects all vias and cells of being camouflaged. This excessively increases the difficulty of reverse engineering. Having a clear idea of the stealthiness of locking/camouflaging nano-primitives is key to an accurate modeling of attackers.

Another critical dimension in modeling attackers is their intent. What is the goal of the attacker? Is it to learn the exact functionality of the circuit or just approximate it? We can enumerate the following with respect to this dimension:

- **Exact Functional Recovery**: The attacker wants to learn the exact Boolean function of c_o. If the attacker recovers a circuit that misbehaves on only a single input pattern, then he/she considers the attack a failure.
- **ϵ-Approximate Functional Recovery**: The attacker intends to approximate the functionality of the circuit with $(1 - \epsilon)$-accuracy, i.e, recover function h that agrees with c_o on a $(1 - \epsilon)$ portion of the input space.
- **Functional/Structural Identification**: The attacker wants to identify and categorize circuit blocks (e.g., find multipliers, adders, and control units), find a specific elements such as the instruction register, or find the boundary of modules or particular wires, and so on.

We digress here to discuss a pitfall in obfuscation research. In addition to the above attacker intents, there is another attacker intention that applies only to locking. That is "an end-user attacker attempting to find a key that will unlock a pristine fabricated locked IC." It can be shown that focusing solely on such an attacker intent with locking may not be reasonable. This is because achieving such a goal is not related to functional obfuscation at all. The defender can simply implement a key-comparison circuit onto every chip that works along with a physically-unclonable-function (PUF) to check for a unique key before activating the power-line or a critical module in the IC. This way an end-user cannot "activate" a fabricated pristine IC without knowledge of the unique key, even though the circuit is not obfuscated (protected against functional reverse engineering) at all. Hence, the more interesting goal of locking is not to prevent end-users from activating chips; rather, it is to obscure the Boolean functionality of the circuit by intelligently and intimately mixing the circuit with a secret key in a non-removable way. This stronger notion of functional obfuscation implies the difficulty of unauthorized activation but the reverse is not true. This point has implications for protections schemes that we discuss in Section 5.

As for the three functional attacker intents defined above, a majority of attacks and defenses fall into the first category. AppSAT [80], DDIP [84], the hill-climbing attack [56], and approximation-resiliency relate to the second class. At this point in obfuscation research we do not have a formal definition and analysis of the third category of functional identification. In other words, we do not know how hard it is to detect that an obfuscated multiplier is still a multiplier without having to deobfuscate it first. The same goes for finding specific wires and modules in an obfuscated circuit. This is in part due to the difficulty of functional identification on unobfuscated circuits in the first

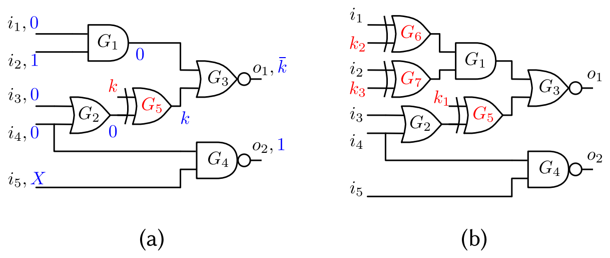


Fig. 4. Example for the testing-based attack: (a) Easily resolvable netlist and (b) non-resolvable netlist.

place, even though the area is seeing some advances [30, 45, 88]. The remainder of the section discusses existing attacks fitting the various above categories in more detail.

3.3 Oracle-guided Combinational Attacks

3.3.1 Testing-based Attack. The oracle-guided attack model was first used in the testing-based attack by Rajendran et al. [60]. The attack leverages circuit testing principles, including justification and sensitization. In the justification process, the output of a gate can be forced to a known value by controlling one or more of its inputs. For example, an OR gate's output can be justified to 1 by setting one of its inputs to 1. In the sensitization process, a net can be observed at the circuit primary output through a certain path, which can be realized by setting all the side inputs of each gate along the path to the non-controlling value of the gate.

Following the unified key-based model of an obfuscated circuit, the testing attack leaks the value of key bits by sensitizing the output of the circuit to a single key bit. To leak this key bit to the output, the gates on the path that goes from this key bit to the output need to be made sensitive to the key bit. This requires setting the side inputs on these gates to non-controlling values. For example, consider the circuit in Figure 4(a). By setting i_3 and i_4 to 0 and 0, respectively, the input to the key-gate G_5 is justified to 0. Meanwhile, by setting i_1 and i_2 to 0 and 0, respectively, the output of the key-gate G_5 can be observed at the primary output o_1. Hence, after applying the input vector $\{0000X\}$, if the output vector is $\{01\}$, then, we get $k = 1$; otherwise, we have $k = 0$.

It is simple to see that in the testing-based attack as soon as there are two key bits on a given leakage path, where one of the key-bits cannot be suppressed with a chosen input pattern, the output will depend on the value of both key bits. The original testing-based attack resorted to a brute-force enumeration when faced with such multi-key paths. As such, the testing-based attack is capable of reasoning only with input-output equations that have one unknown. One example of the non-resolvable netlist is shown in Figure 4(a). We discuss defenses against such weak key bits later in the article.

3.3.2 The SAT Attack. The SAT-based attack or SAT attack proposed in References [26, 87] significantly enhances the deobfuscation efficiency. Unlike the testing-based attack it can work with systems of input-output equations with an arbitrary number of key bits. The SAT attack iteratively searches for input patterns that are guaranteed to prune incorrect key combinations and has a termination criterion that guarantees that if the original hypothesis space C includes a correct functionality it will find it within a finite amount of time.

Recall the example in Figure 4(a) and assume the correct key is 0, 1, 0 for k_1, k_2, k_3, respectively. Since the sensitization and justification condition cannot be satisfied for each key bit, the brute-force attack has to be exploited, which requires at least $2^3 = 8$ input patterns to prune all the incorrect key combinations [26]. However, with the SAT attack, as shown in Figure 5, only three input patterns are sufficient. As per Figure 5, the SAT attack at any point in the process only

	Input Patterns				
Key Vectors	$(1,1,0,0,X)$	$(1,0,0,0,X)$	$(0,0,0,0,X)$	$(1,0,1,0,X)$	$(0,0,1,0,X)$
$(0,0,0)$	$(0,1)$ ✗	$(1,1)$	$(1,1)$	$(0,1)$	$(0,1)$
$(0,0,1)$	$(1,1)$	$(0,1)$ ✗	$(1,1)$	$(0,1)$	$(0,1)$
$(0,1,0)$	$(1,1)$	$(1,1)$	$(1,1)$	$(0,1)$	$(0,1)$
$(0,1,1)$	$(1,1)$	$(1,1)$	$(0,1)$ ✗	$(0,1)$	$(0,1)$
$(1,0,0)$	$(0,1)$ ✗	$(0,1)$	$(0,1)$	$(1,1)$ ✗	$(1,1)$ ✗
$(1,0,1)$	$(0,1)$ ✗	$(0,1)$	$(0,1)$	$(0,1)$	$(1,1)$ ✗
$(1,1,0)$	$(0,1)$ ✗	$(0,1)$	$(0,1)$	$(1,1)$ ✗	$(1,1)$ ✗
$(1,1,1)$	$(0,1)$ ✗	$(0,1)$	$(0,1)$	$(1,1)$ ✗	$(0,1)$

Fig. 5. Outputs for all possible key combinations for the netlist in Figure 4(a) for four input patterns (assume the correct key bits are $(1,1)$). The red "X" marks show input patterns ruling out incorrect keys.

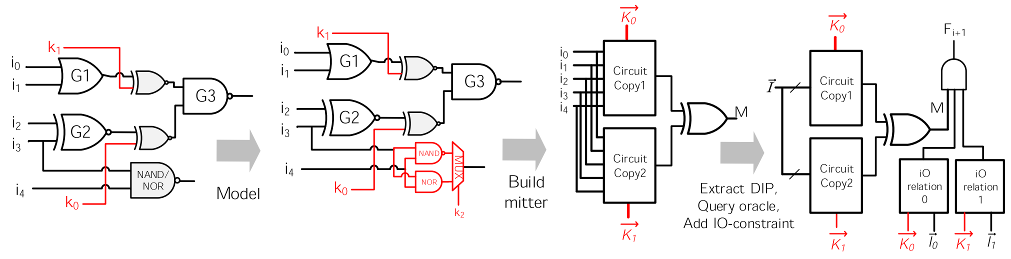


Fig. 6. The SAT attack [26] flow can be viewed as a set of transformations steps applied to a circuit that represents a Boolean condition that will be refined to eventually resolve to a correct key.

picks input patterns (columns) that have disagreeing keys (disagreeing rows). This ensures that no redundant query is made. As such the attack first picks $\{1,1,0,0,X\}$ and $\{1,0,0,0,X\}$ to query, which returns $(1,1)$ and $(1,1)$. Any key combination that produces an incorrect output given the patterns queried so far is pruned. Removing the rows that disagree with these observations, we can move on to the next query that has disagreeing rows, which would be $\{0,0,0,0,X\}$ rather than $\{0,0,1,0,X\}$ or $\{1,0,1,0,X\}$.

Enumerating such an attack table explicitly will be awfully inefficient. The SAT attack avoids this by representing this table using concise Boolean formula/circuits. Figure 6 shows this process. The first step is to model the ambiguity in the netlist with key-variables, i.e., move to a locking formulation. Then a "mitter" circuit is built by copying the circuit and then connecting the output of the copies to a comparator to create the mitter output $M = (c_e(i,k_1) \neq c_e(i,k_2))$. The attack works with the circuit/formula F_0 that is initialized to M. It is easy to see that M will be true if there exists $\hat{i}$, $\hat{k}_1$, and $\hat{k}_2$ such that the obfuscated circuit produces a different output for $\hat{k}_1$ versus $\hat{k}_2$ when the input is $\hat{i}$.

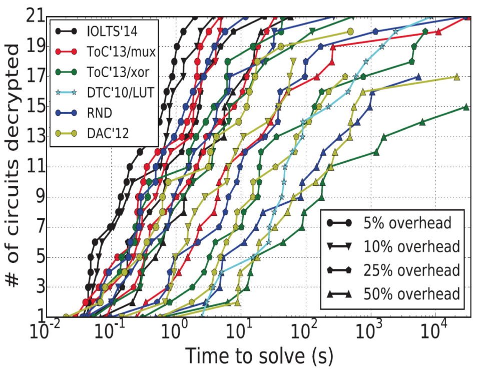
The attack hence begins by satisfying F_0 with $\hat{i}_0$, $\hat{k}_1$, and $\hat{k}_2$. This $\hat{i}_0$ is referred to as a discriminating input pattern (DIP). The DIP is queried on the oracle, and we get $\hat{y}_0 = c_o(\hat{i}_0)$. Observing the correct output $c_o(\hat{i}_0)$, at least one of $c_e(\hat{i}_0, \hat{k}_1)$ and $c_e(\hat{i}_0, \hat{k}_2)$ should be different from $c_o(\hat{i}_0)$. Hence, $\hat{i}_0$ is guaranteed to prune at least one incorrect key combination. Hence adding this input-output condition to F_0 will encode this information:

$$F_1 = F_0 \wedge (c_e(\hat{i}_0, k) = c_o(i_0)) \wedge (c_e(\hat{i}_0, k_0) = c_o(\hat{i}_0)).$$

Now by satisfying F_1 we are searching for a new DIP that can further prune the space of keys that already satisfy $(\hat{i}_0, c_o(\hat{i}_0))$. Such a process can be continued until F_i is no longer satisfiable. At this

Algorithm 1. Given oracle access to c_o and the expression for c_e return a correct key $k_1 \in K_*$.

```
 1: function SATDECRYPT($c_e$, $c_o$ as black-box)
 2:     $j \leftarrow 0$
 3:     $M \leftarrow c_e(i, k_1) \neq c_e(i, k_2)$
 4:     $F_j \leftarrow true$
 5:     while $F_j \wedge M$ is solvable do
 6:         Solve $F_j \wedge M$ with $\hat{i}_j, \hat{k}_1, \hat{k}_2$
 7:         $o_j \leftarrow c_o(\hat{i}_j)$
 8:         $F_{j+1} \leftarrow F_j \wedge (c_e(\hat{i}_j, k_1) = o_j) \wedge (c_e(\hat{i}_j, k_2) = o_j)$
 9:         $j \leftarrow j + 1$
10:     end while
11:     satisfy $F_j$ with $k_1$ and $k_2$
12: return $k_1$ as exact key
13: end function
```



(a)

Fig. 7. Algorithm 1: The SAT attack algorithm. (a) Runtime for 21 different benchmark circuits with six different locking algorithms and four different overhead levels. Panel (a) is courtesy of Reference [75].

(a)

	i^0	i^1	i^2	i^3	i^4	i^5	i^6	i^7
k^*	✓	✓	✓	✓	✓	✓	✓	✓
k^1	✗	✓	✓	✓	✓	✗	✗	✗
k^2	✓	✓	✓	✓	✗	✗	✗	✗
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k^{13}	✓	✗	✗	✓	✗	✗	✓	✓
k^{14}	✓	✗	✗	✗	✓	✓	✗	✓
k^{15}	✗	✓	✗	✓	✗	✓	✗	✓

(b)

	i^0	i^1	i^2	i^3	i^4	i^5	i^6	i^7
k^*	✓	✓	✓	✓	✓	✓	✓	✓
k^1	✗	✓	✓	✓	✓	✓	✓	✓
k^2	✓	✗	✓	✓	✓	✓	✓	✓
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k^{13}	✓	✓	✓	✗	✓	✓	✓	✓
k^{14}	✓	✓	✓	✓	✓	✗	✓	✓
k^{15}	✓	✓	✓	✓	✓	✓	✓	✗

(c)

	i^0	i^1	i^2	i^3	i^4	i^5	i^6	i^7
k^*	✓	✓	✓	✓	✓	✓	✓	✓
k^1	✗	✓	✓	✓	✓	✓	✓	✓
k^2	✓	✗	✓	✓	✓	✓	✓	✓
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k^{13}	✓	✗	✗	✓	✗	✗	✓	✓
k^{14}	✓	✗	✗	✗	✓	✓	✗	✓
k^{15}	✗	✓	✗	✓	✗	✓	✗	✓

Fig. 8. The truth tables for three different obfuscated circuits with four key bits and three inputs: (a) high-error obfuscation, (b) low-error obfuscation, and (c) mixture of high-error and low-error obfuscation.

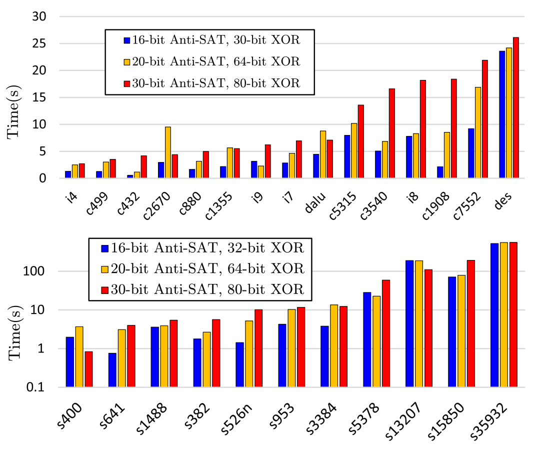
point, we can remove M from F_i and only solve the I/O-constraints. This should return a correct key if the c_e included a correct key to begin with. Algorithm 1 shows this flow.

Although SAT is in general an NP-complete [24] problem, SAT problems that are deduced from Boolean circuits can be solved quite efficiently with modern SAT solvers [24]. For traditional obfuscation schemes such as EPIC, the number of DIP queries that the SAT attack requires is extremely small compared to the brute-force complexity. Figure 7(a), shows SAT attack runtime results for a collection of 21 ISCAS and MCNC benchmark circuits obfuscated with various traditional locking schemes. As can be seen the SAT attack is capable of resolving the correct key within a matter of minutes to hours. It was observed in Reference [87] that the average number of discriminating inputs is less than 50 and the average attack time is less than 2500s, which is in stark contrast to the estimates for a brute-force attack that may require millions of years to finish.

3.3.3 Approximate Attacks. The SAT attack described above is an exact attack. It terminates only when no more DIPs can be found, which implies that the recovered key k_* if programmed into c_e will produce a perfectly equivalent circuit to c_o. This works well for the traditional high-error obfuscation approaches that randomly insert key-structures into the circuit. However, the exact SAT attack is agnostic to the error rate of the obfuscation, which can cause problems when faced with low-error rate obfuscation. Consider the example in Figure 8. Figure 8(a) shows the truth table for an obfuscated circuit generated with a high-error scheme. Because the output error probability

Algorithm 2. Given oracle access to c_o and the expression for c_e return an approximate key k_1.

```
1:  function ApproxSATDecrypt($c_e$, $c_o$ as black-box)
2:      $j \leftarrow 0$
3:      $M \leftarrow c_e(i, k_1) \neq c_e(i, k_2)$, $F \leftarrow true$
4:      while $F \wedge M$ is solvable or term do
5:          Solve $F \wedge M$ with $\hat{i}$, $\hat{k}_1$, $\hat{k}_2$
6:          $\hat{o} \leftarrow c_o(\hat{i})$
7:          $F \leftarrow F \wedge (c_e(\hat{i}, \hat{k}_1) = \hat{o}) \wedge (c_e(\hat{i}, \hat{k}_2) = \hat{o})$
8:          every $d$ rounds do
9:          term $\leftarrow$ AnalyzeError($F$, $k_1$, $c_e$, $c_o$)
10:         term $\leftarrow$ CheckTermination($F$, $c_e$, $c_o$)
11:         WireDisagreement($i_j$, $k_1$, $k_2$, $c_e$)
12:     end while
13:     satisfy $F$ with $\hat{k}_1$, $\hat{k}_2$
14: return $\hat{k}_1$ as key
15: end function
```



(a)

Fig. 9. Algorithm 2: AppSAT algorithm enhanced with wire-disagreement analysis and termination checking. (a) Runtime of the AppSAT attack on combinational and sequential benchmarks on compound schemes (Anti-SAT + EPIC). The figures are adapted from Reference [75].

for each incorrect key is large, the number of incorrect keys pruned by one discriminating input can be very large. For example, in Figure 8(a), the input pattern i^6 can prune four incorrect keys. However, when the output error probability of the incorrect keys is small, it is less likely for one DIP to disqualify a large number of keys as seen in Figure 8(b) where each discriminating input, e.g., i^1, can only prune one incorrect key.

AppSAT was first proposed to allow the SAT attack to keep track of the error-rate of intermediate keys. Compared with the exact SAT attack described in Algorithm 1, AppSAT follows the same steps including searching for and storing DIPs. The main difference comes from the output error probability estimation step. In AppSAT, after every fixed number of iterations, a key satisfying all the existing input-output patterns is extracted from the SAT-solver. Then, the output error probability of this key is estimated through random sampling. If the error probability is larger than the given threshold, then AppSAT continues to search for the discriminating inputs. Otherwise, the algorithm just stops and returns the key.

One important idea in AppSAT is that during the random sampling step, new DIP's are discovered every time the oracle and c_e disagree on a random input. Some of these DIPs are also stored in back into the SAT problem. This causes the high-error keys to be pruned first. For example, in Figure 8(c), k^1 and k^2 have low error probabilities while k^{13}, k^{14}, and k^{15} have high error probabilities. In the first iteration of AppSAT, all the input patterns except i^7 are discriminating inputs. Then, when one input is randomly selected from i^0 to i^6, the probability for k^{13}, k^{14}, and k^{15} to be pruned is 4/7, which is much higher than the probability for k^1 and k^2 to be pruned. Because of this property, that keys with low error probabilities can be obtained within a small number of iterations.

AppSAT can be used to attack compound obfuscation schemes that use a combination of low-error schemes plus a high-error scheme. Figure 9(a) shows the deobfuscation time for AppSAT evaluated on the ISCAS and MCNC benchmarks obfuscated with such compound schemes. Double-DIP [84] and the bypass attack [101] followed AppSAT with techniques to target specific compound schemes. DDIP added a property to the SAT formula that causes the attack to mine for DIPs can disqualify at least two incorrect keys allowing it to terminate only when the low-error scheme is left to recover. In Reference [80] similar termination conditions were further studied and added to AppSAT. In addition, AppSAT was later augmented with removal attacks [106] and wire-disagreement analysis [80], which are tools that use the intermediate error rate of the hypothesis for locating hot-spots in the circuit during the deobfsucation process. Algorithm 2 shows AppSAT with the various intermediate analysis steps. DDIP has a flow similar to the original SAT attack in Algorithm 1 except the mitter M has 4 or more keys rather than 2. In Reference [5], a satisfiability-modulo-theory (SMT)-solver to increase the number of incorrect keys that each DIP disqualifies to speed up process.

Another attack that can be categorized as an approximate attack and was proposed before the SAT attack's invention, is the hill-climbing attack [56]. This attack defines a key hypothesis k_h along with a best key hypothesis k_{bh}. The key bits in k_h are considered as parameters in an optimization problem where the cost function is the disagreement probability between $c_e(i, k_h)$ and $c_o(i)$. Since the exact disagreement probability cannot be calculated efficiently it is estimated with random sampling similar to AppSAT. in each iteration, k_h is derived by making changes to k_{bh}. Then the error rate for k_h is estimated, and if the error rate is better than the error rate of the best key hypothesis k_{bh}, then k_h becomes the new k_{bh}. The hill-climbing attack uses simulated-annealing for deriving k_h from k_{bh}. It first derives a bit-flip probability using the simulated-annealing formula and then flips bits in k_h according to this probability. This results in bits being flipped at a higher rate in the beginning of the attack when the error rate for k_{bh} is high. As the error rate drops and we get a better key hypothesis the flip rate drops exponentially finally converging on a low error key. A great advantage of the hill-climbing attack is that it does not rely on NP-complete SAT queries like SAT attacks.

3.4 Oracle-Less Attacks

The oracle-less (OL) model assumes that the attacker only has access to the netlist for c_e and knowledge of the obfuscation scheme used. Due to this constrained set of resources that the attacker has, oracle-less attacks are quite attractive. The oracle-less attacker has to study the small statistical variations in the structure and functionality of the obfuscated circuit itself to recover the original circuit.

So far oracle-less attacks have focused on logic locking instead of camouflaging as the OL model is more likely in the locking domain. One major theme for such attacks is to study the vicinity of the inserted key structure, since key-logic insertion typically effects only the locality of the insertion site, which we refer to as a circuit cut. Then various key possibilities are enumerated for this circuit cut and then the cut is simplified under each key. It turns out that many such key possibilities when simplified result in cuts that can be pruned given general known properties/rules of high performance circuits. For instance, El massed et al. [51], who proposed the first oracle-less attack used the fact that the simplified cut should not deviate vastly from the original circuit minus the key-structure to prune such improbable key assignments. In Reference [40], the fact that some keys produce circuits with low testability is used to prune these keys, since the original circuit is typically synthesized to be highly testable.

Another idea is to resort to statistical analysis and machine learning. The original circuit or circuit cut has statistics $P(c_o)$ and the obfuscated circuit has statistics $P(c_e)$. Ideally, these statistics should be independent meaning that $P(c_e) = P(c_e|c_o)$. In practice, however, schemes that simply insert fixed key-structures into the circuit with limited post-insertion randomization will demonstrate dependent statistics. This means that given $P(c_e)$ itself with no additional information, the space of possible c_o's shrinks. These statistics can be mined with various statistical inference tools, including machine-learning schemes, and then reversed. Chakraborty et al. carried this out in Reference [12] against random XOR/XNOR insertion with a success key bit recovery rate of around 90% on a subset of ISCAS benchmarks.

Algorithm 3. Given sequential-oracle access to c_o and the expression for the sequential c_e return an exact key k_1.

```
1: function SeqDecrypt($c_e$, $c_o$ as black-box)
2:     $j \leftarrow 0, b \leftarrow 1$
3:     $M \leftarrow c_e(i, s_1, k_1) \neq c_e(i, s_2, k_2)$
4:     $F_j \leftarrow true$
5:     while !Termination($F_j$) do
6:         if Bmc($F_j \wedge M$, G($\neg M$), $b$) $\rightarrow Fail$ then
7:             $\hat{O}_j \leftarrow c_o(\hat{I}_j)$    $\hat{I}_j \leftarrow$ Cex(G($\neg M$))
8:             $F_{j+1} \leftarrow F_j \wedge (c_e^b(\hat{I}_j, k_1) = \hat{O}_j) \wedge (c_e^b(\hat{I}_j, k_2) = \hat{O}_j)$
9:             $j \leftarrow j + 1$
10:        else
11:            $b \leftarrow b * 2$
12:        end if
13:    end while
14:    satisfy $F_j$ with $k_1$ and $k_2$
15: return $k_1$ as key
16: end function
```

B'Mark	# Disc Inputs		Max Steps		Time (s)		Termination
	min	max	min	max	min	max	UC/CE/UMC
s344	3	5	10	10	11	37	10/0/0
s349	3	7	10	10	15	69	10/0/0
s382	25	36	50	60	3482	41129	10/0/0
s400	18	34	50	90	4921	526499	6//0/1
s444	16	35	50	90	3379	52984	2/0/5
s510	7	15	30	40	300	29121	10/0/0
s526	29	39	120	120	37979	139252	10/0/0
s820	14	20	10	10	506	1030	10/0/0
s832	12	21	10	10	370	1211	10/0/0
s953	10	22	10	10	365	1709	10/0/0
s1196	14	44	10	10	795	2386	10/0/0
s5378	7	-	30	-	1350	-	0/1/0
s9234	-	-	-	-	-	-	0/0/0
b04	4	9	10	10	31	151	10/0/0
b08	26	117	20	20	619	10527	10/0/0
b14	14	21	10	10	14308	34273	10/0/0

Fig. 10. Algorithm 3: The model-checking attack algorithm for deobfuscation under the sequential-oracle-guided (SOG) attack model. Line 6 makes a call to a BMC solver with the $G(\neg M)$ property. Line 5 checks for termination conditions such as checking if the next-state and output functions are unique under the current set of DIPs. The table to the right shows data on ISCAS sequential benchmarks. The step column represents the depth of the queries (number of clock cycles). The table is adapted from Reference [25].

3.5 Sequential Oracle-guided Attacks

The oracle-guided (OG) attack framework can be extended to the sequential oracle-guided attack (SOG) model discussed earlier. In a sequential attack model the attacker cannot observe/control the internal state of the oracle c_o. The attacker can reset the IC and then pass it inputs and observe outputs. Meade et al. [52] first proposed a method for extending the SAT attack to sequential circuits via unrolling. The idea is to simply unroll the sequential c_e up to b cycles. This unrolled circuit denoted as $c_e^b(\mathrm{I}, r, k)$ now has b input vectors I and b output vectors. The latch inputs in cycle i are connected to latch outputs in cycle $i + 1$ except for the first cycle latches, which are initialized with the reset state r. This unrolled circuit is a combinational circuit, and hence the SAT attack can be directly replicated. An unknown initial state can be modelled by additional virtual key variables.

Checking Boolean properties by unrolling a circuit and passing it to a SAT solver has been the topic of research in the verification domain for years, and it is known as the Bounded-model-checking (BMC) problem for which an array of tools and techniques exist. El Massad et al. [50] presented the first in-depth analysis of sequential attacks using a model-checking formulation. They presented several termination criteria for knowing when to exit the sequential attack, a detail that was missing in Meade et al.'s [52] work. They showed how the attack runtime increases drastically for circuits that do not leak their internal state to the output with high error. Yet it was observed that very large sequential circuit can be doebfuscated with only mildly more runtime than the combinational SAT attack. The sequential/model-checking attack algorithm along with El Massad et al.'s data from Reference [50] is shown in Figure 10. In Reference [79], the model-checking attack runtime was improved drastically by dynamic simplification of the circuit conditions when unrolling. Before the arrival of these attacks, several works [32, 33, 74] proposed scan-chain deactivation/subversion/obfuscation as a way to secure locking against oracle-guided attacks. While limiting scan-chain access is a low-cost way to significantly improve security overall, smaller obfuscated circuits with high-entropy/low-depth behavior may still be vulnerable to sequential attacks.

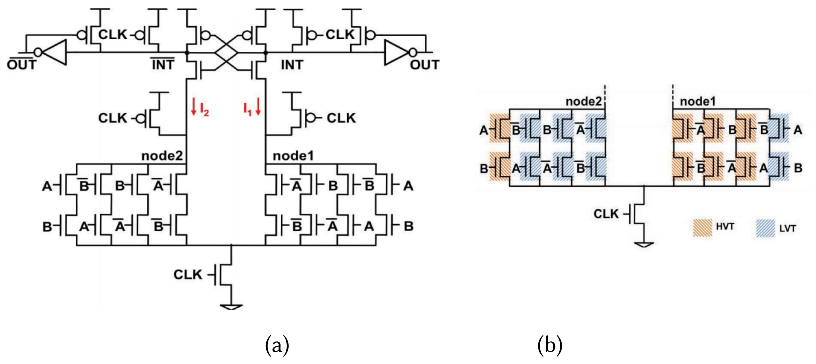


Fig. 11. Two-input threshold voltage defined gate design for different functionalities: (a) schematic and (b) an example for an AND gate. The figures are adapted from Reference [27].

4 ON DEFENCES

In this section, we review locking/camouflaging schemes themselves. We start by discussing the nano-technology aspects of these techniques and then delve into the netlist-level algorithms used to insert such structures into the original circuit. Along the way, we also discuss attacks that are more specific to the particular obfuscation scheme.

4.1 Camouflaging Cell/Device-level Primitives

A camouflaged nano-device is a silicon-level primitive that creates ambiguity for a microscopy-based reverse engineering process. Such structures can be implemented by special fabrication flows. We review some of these techniques herein. Note that there are theoretically many ways for creating ambiguous nano-structures; however, they should be evaluated on the basis of metrics such as manufacturing cost overhead (in terms of added masks and steps), resiliency to reverse engineering techniques, and footprint size per the amount of ambiguity that they create. It is difficult to completely isolate the nano-device from the cell that uses it and how the cell is incorporated in the netlist. Therefore, we will discuss some cell designs along with the nano-device.

One major approach to building camouflaging structures is to utilize doping levels [27]. One way to use doping is to create gates that have similar metal and poly patterns but with different doping patterns [20, 42, 85]. In addition, the doping level of a transistor affects its on/off-resistance, and this difference in resistance can be detected with a sense-amplifier circuit. An example of this is shown in Figure 11(a). In this cell when the resistance of the right branch is more than the left branch through selecting the doping levels of the transistors in each branch, the sense amplifier can implement an array of functions. The doping level of the 16 different transistors can allow implementing all 16 different two-input Boolean functions. The cell introduces a particularly high overhead. For example, when the cell functions as an AND gate, compared with the existing design, it introduces 70% delay overhead and 160% area overhead.

Unfortunately, camouflaging based on doping types (p-type versus n-type) cells are not sufficiently secure against Scanning-Electron-Microscope (SEM) and Focused-Ion-Beam (FIB) imaging. Since the doping level in the substrate interacts with the electron-beam coming from the SEM, by varying the intensity level of the electron-beam, the doping of the substrate can be differentiated. Figure 12 shows such a doping-type camouflaged circuit under an optical microscope, SEM and FIB. Smaller variations in doping intensity (lightly n-doped versus heavily n-doped) may be harder to detect under SEM/FIB [73].

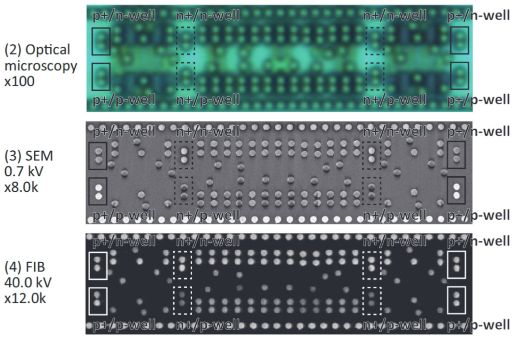
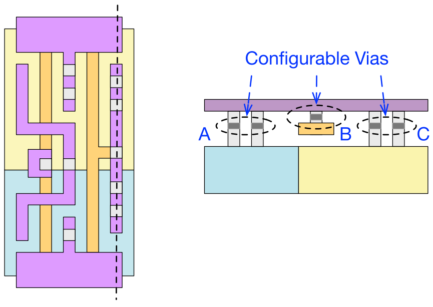


Fig. 12. Optical, SEM, and FIB imaging of a doping-based camouflaged cell reveal its functionality. Image courtesy of Reference [89].



Func.	BUF	INV
Timing	0	100%
Area	0	50%
Power	0	50%

(a) (b) (c)

Fig. 13. Example of the camouflaging cell design proposed in Reference [43]: (a) top view, (b) cross section, and (c) overhead for different functionalities.

A more secure and light-weight approach to building camouflaging primitives is based on dummy contacts/vias. A dummy contact is a via that appears similar to a normal via from the top view but is in fact non-conducting. This can be implemented by a via with a middle-gap/insulator or a via with a non-conducting material such as MgO as proposed in Reference [15]. This can be used to build camouflaging cells or used to obfuscate interconnects by adding additional wires and dummy vias to the circuit. An example of a camouflaging cell utilizing such vias is shown in Figure 13. Depending on the connectivity of the vias in group *A*, *B*, and *C*, the cell can work as an INV or a BUF. The overhead of the cell depends on its functionality as is shown in Figure 13(c). Dummy vias may be able to achieve the ubiquitous ambiguity notion discussed in Section 3.2, where the attacker may have to suspect all vias in the design as being fake.

4.2 Locking Cell/Device-level Primitives

Logic locking unlike camouflaging requires a nano-device that is *programmable*, as programmablity is the foundation of logic locking. There are various approaches to programming key-bits into a locked circuit and different approaches can in fact have vastly different security implications [3, 107]. This is why it is critical to consider the programmable device technology when analyzing netlist-level security.

The first and simplest approach to programming keys is to use conventional volatile CMOS circuits such as a scan-chain of latches/DFFs. In this case, the programmable logic will resemble the rest of the CMOS circuit and can potentially create a scenario in which the attacker does not know which parts of the circuit are secret keys. However, since the key logic is volatile, the key may have to be programmed on every start-up of the chip. Hence, it is easy to see that some sort of nonvolatile memory is necessary.

At this point, a critical question of **activation** in a locking flow arises [107]: *Who and at which point in the supply chain will program the locking key into the device?* The answer to these questions have security implications and are in fact tied to the question of the cell/device used in locking. There are a few possible scenarios. If the end-user is to program the secret key, then he/she can use an external nonvailte memory to store the key. This leaks the key to the end-user, which if he/she collaborates with the foundry will violate protections against the foundry. The key can be programmed by a trusted system designer (PCB designer and assembler), which places the external nonvolatile memory on the PCB and programs the key and sells the system to an untrusted end-user. In this case, if we are to hide the key from the end-user, then the nonvolatile external memory will have to be tamper-resistant plus the board-level communications will have to be secured with encryption, and so on. A collaborative attack between the system designer and the foundry will again violate security against the foundry.

It is easy to see that the activator party can almost always violate the protection scheme if this party is untrusted and collaborates with the foundry. One may think of avoiding this by releasing an encrypted version of the key to the activator and have the fabricated IC decrypt the encrypted key into an internal secret key. This, however, will require another secret string used for decryption to be stored on the IC, which itself will have to be programmed by a trusted party into a nonvolatile tamper-resistant medium. Furthermore, to prevent one key from unlocking all the other chips a unique per chip signature is required, which calls for a Physical-Uncloanable-Function (PUF). Again, a trusted facility will be required to characterize this PUF post-fabrication. The PUF and key decryption infrastructure may be removed or tampered with by the foundry as well.

Given the above discussions it becomes clear that a trusted activator party along with a non-volatile tamper-resistant storage are unavoidable in a locking-enhanced supply chain. Plus, having this nonvolatile storage on-chip as opposed to off-chip will avoid the many issues associated with an off-chip key-storage that brings us to the point that an on-chip tamper-resistant nonvolatile technology is critical to secure and low-overhead locking. We discuss some of these technologies herein.

Embedded Flash and EEPROM based on floating-gate devices are a good candidate for key-storage [31, 36, 38]. These devices can be placed in an array and read-out using a sense-amplifier and then fed to a scan-chain of key-bits. Another approach is to integrate them directly into the cells as multi-level resistors that can be read locally with an amplified tree structure similar to Figure 11(a). Embedded Flash is currently an industry standard and scales well into deep submicron technologies. The state of Flash and EEPROM cells can be recovered with tedious reverse engineering processes using controlled electron-beams [23], but this becomes more and more difficult as the technology scales.

Another candidate for locking is Resistive-RAM (RRAM). Metal-insulator-metal (MIM) structures can be used to implement programmable resistances also known as memristors. If the R_{on}/R_{off} ratio of these cells is high and they have a low R_{on}, then they can be placed directly on the signal path [21]. The cell shown in Figure 14 leverages this to implement an INV/BUF cell. Magnetic-Tunneling-Junctions (MTJs) can also implement double-state resistors. However, they have a much smaller R_{on}/R_{off} with high R_{on} levels. This makes it difficult to insert them onto signal paths and instead necessitates sense-amplifier-based reading of their value. The value of MTJs can also be read out with magnetic-force-microscopy (MFM). If the area penalty for using a sense-amplified tree are acceptable to a designer, then CMOS and FinFET devices themselves can be used

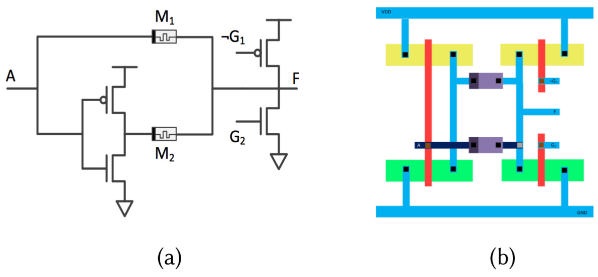


Fig. 14. Camouflaging cell design leveraging memristors: (a) schematic of the cell that can function as an INV or a BUF; (b) the cell layout. The figures are adapted from Reference [65].

as programmable resistors through intentional hot-carrier-injection (HCI) as done recently in Reference [2].

Fuse and Anti-fuse technology are one-time-programmable (OTP) devices. These devices can be very suitable for locking, since in locking we typically do not require multi-programming ability. There are various fuse/anti-fuse device designs with different footprints and security levels. Planar fuses, for instance, may not be secure as their state is visible from the top view due to electromigration. Metal-to-metal antifuse [22] is a great device for interconnect obfuscation, as it takes no substrate space, can have a layout similar to a normal via, and its state is very difficult to determine under microscopy [10, 22]. Fuse technology in advanced nodes seems to be moving toward using the gate-oxide in CMOS and FinFET transistors as fuses themselves [1], which may be due to scaling issues with metal-to-metal antifuse. Given the unique benefits that such metal-to-metal antifuse brings to locking as shown in Reference [78] it may be reasonable for the industry to reinvest in deeply scaled and reliable metal-to-metal programmable switches.

4.3 Netlist-Level Obfuscation Schemes

Given the original netlist and the basic obfuscation building blocks, a defender must synthesize a new, obfuscated netlist to maximize the resilience against deobfuscation at minimum power, performance, and area cost. In this section, we review the following representative netlist-level schemes, including random gate insertion [69], the clique-based scheme [59, 61], point-function schemes [43, 99, 105], cyclic obfuscation schemes [77, 78], FSM transformation, and timing-based/parametric obfuscation schemes [114].

4.3.1 Random Gate Insertion. The first systematic obfuscation scheme, termed EPIC, was proposed in Reference [69]. The main design objective of EPIC was to achieve a large number of possible key combinations and guarantee that only one unique key combination can successfully activate the chip. This design objective can be achieved by randomly inserting key-controlled XOR/XNOR gates into the original circuit [69]. An inserted XOR gate means that the key bit that controls it has to be zero for correct operation. An XNOR creates the opposite. The fact that the correct key bit in XOR/XNOR insertion affects the type of gate that is inserted is itself a source of leakage of secret information that is exploited in oracle-less attacks.

Later on, a look-up-table (LUT) insertion strategy was proposed [8]. In this technique, k-input cuts in the circuit are replaced with k-input LUTs that are programmed with key bits. Since such a cut replacement schemes is a reductive approach (parts of the circuit are removed and replaced), it is much more secure against oracle-less attacks.

4.3.2 Clique-based Scheme. Once the testing-based attack was proposed it was observed that randomly inserting a small number of elements in a large netlist can leave many key bits isolated

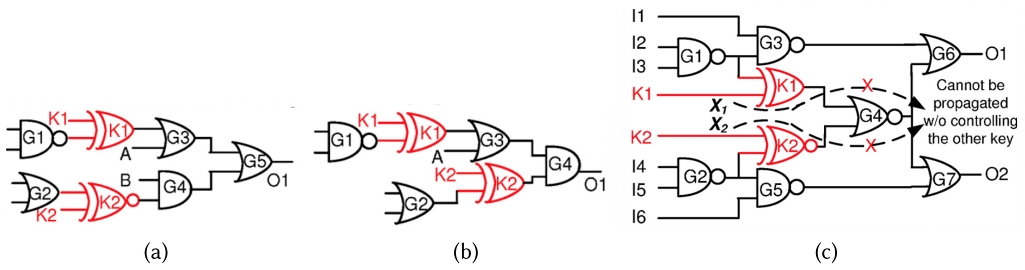


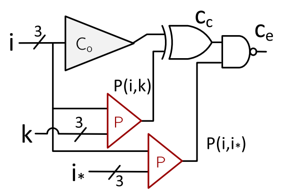
Fig. 15. Example of interfering key-gates: (a) concurrently mutable convergent key-gates, (b) sequentially mutable convergent key-gates, and (c) non-mutable convergent key-gates. The figures are adapted from Reference [59].

and prone to the testing-based attack. As such, Rajendran et al. proposed a clique-based strategy that aimed at ensuring that various key bits are dependent [59, 61]. The clique-based method is based on constructing a key-dependency graph where key bits are nodes that have an edge connecting them if they interfere with one another on a path from a primary input to an output. Cliques in this graph represent sets of keys that all interfere with one another. This notion can be extended [59] to account for the fact that even if two key bits interfere, there may be an input pattern that can mute one of the key bits and isolate the other one. Hence, the authors further classify each pair of interfering key-gates as concurrently mutable convergent key-gates, sequentially mutable convergent key-gates, and non-mutable convergent key-gates. The examples of each category are shown in Figure 15. In Figure 15(a), the two key-gates, i.e., K_1 and K_2, are concurrently mutable as K_1's key bit can be determined by setting the signal B to 0 to mute K_2 while K_2's key bit can be determined by setting the signal A to 1 to mute K_1. Hence, for concurrently mutable key-gates, the attackers can select input patterns that mute any one of the two key-gates and sensitizes the other one. In Figure 15(b), the two key-gates are sequentially mutable as K_2 can be determined by setting the signal A to 1 to mute K_1 while K_1 cannot be determined directly without knowing K_2. For sequentially mutable key-gates, the attackers have to first mute K_1 to sensitize K_2 and then, determine K_1. In Figure 15(c), the two key-gates are non-mutable as the attackers cannot mute either K_1 or K_2. The authors claim that for non-mutable key-gates, the brute-force attack must be conducted and, thus, provide the best resilience against the testing-based attack. Therefore, in the clique-based scheme, the key-gates are inserted iteratively to form a clique, which maximizes the pairs of non-mutable key-gates and increases the resilience against the testing-based attack.

It was later shown in Reference [98] that the computationally heavy clique analysis may not be necessary to create such interference, since inserting a singe key-gate near the output means that no key bit can get through to the output without interfering with at least one more key bit. Plus, since all the keys in this cone converge on each other at the output, they are all interfering with one another.

4.3.3 Fault-based Scheme. Rajendran et al. [62] studied the notion of error rate in obfuscation. They defined the hamming distance of an obfuscation scheme by the average number of output bits that flip as a result of flipping a single key bit. They then proposed using fault-analysis to insert XOR and MUX key-gates in locations that maximize the fault impact, which was a metric describing the degree to which a fault in a particular wire would affect the output. This leads to naturally higher error rates as the circuit's output is more sensitive to changes in these locations.

All the above obfuscation strategies are referred to as traditional schemes as they predate the SAT attack. These schemes have various resiliency levels against oracle-less attacks, and testing-based attacks. They also can have different overhead levels. Yet none of them are able to withstand



input	output	key							
$I_2I_1I_0$	O	K_0^*	K_1	K_2	K_3	K_4	K_5	K_6	K_7
000	0	0	0	0	0	0	0	0	0
001	1	1	0	1	1	1	1	1	1
010	1	1	1	0	1	1	1	1	1
011	0	0	0	0	1	0	0	0	0
100	0	0	0	0	0	1	0	0	0
101	1	1	1	1	1	1	0	1	1
110	0	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	0	1

Fig. 16. A point-function scheme example. The truth table on the right demonstrates that a SAT attack will have to query all input patterns to disqualify all possible keys (red cells). Figure and table are adapted from Reference [80].

the SAT attack with low overhead on decent sized combinational circuits. Subramanyan et al. [87] showed how all the above schemes when applied to the ISCAS benchmarks with thousands of gates with overhead values bellow 25% can be attacked in a matter of minutes to hours. Note that all these schemes also produce acceptable error rates.

4.3.4 *Point-Function-based Schemes.*

The first SAT-resilient locking/camouflaging schemes were based on using point-functions [43, 99, 105, 108]. A point-function (essentially a comparator circuit) is a function $P(x, x_*)$ that outputs 1 when $x = x_*$ and 0 otherwise. Such a comparator can be implemented with an AND-tree with inputs that are XORs of bits in x and x_*. By incorporating these point-functions into a circuit in specific ways it is possible to create obfuscated circuits that need at least an exponential number of queries to be exactly resolved. An example of such a point-function scheme along with its two-dimensional truth-table can be seen in Figure 16. It can be seen from this figure how two point-functions can be used to create a low-error obfuscation that requires the majority of the input space to be queried to recover the exact functionality.

While the tradeoff between error rate and query count seems intuitive, one question is if it can be described by tight bounds. In Reference [115] this tradeoff was modeled as a matrix cover problem and an upper bound on the product of error rate and minimum query count was obtained. It was later shown in Reference [80] how this bound is inaccurate as there exists schemes that surpass this bound and finding the accurate bound in special cases can be reduced to an open problem on regular hypergraphs.

Li et al. [43] proposed a point-function scheme based on searching in the original circuit itself to find a naturally occurring AND/OR-tree and obfuscate it by inserting XOR-gates at its inputs in a similar way. The approach in Reference [43] is quite attractive for circuits that naturally have such point-functions in their structure; however, this may not be true in the general case. Anti-SAT was the first point-function scheme that added such structures into the circuit [99]. By taking the AND of two point-functions with two different key vectors (k_1 and k_2) as seen in Figure 17(b) an exponential minimum DIP count is achieved. References [80, 115] included tree-based schemes and Reference [104] presented a hamming-distance-based block to increase the error-rate of such SAT-resilient schemes at the cost of drops in query count with various tradeoff rates. For example, in Reference [80] error rate increases with 3^t while minimum query count drops with 2^t. In general, if function $h(i, k)$ is used to corrupt the output, where the onset size of $h(i, k)$ for each fixed k is M, then one can expect a minimum query count of at least $2^n/M$ [80, 115].

One main challenge with point-function schemes is their vulnerability to removal attacks [106]. Since Anti-SAT and Li et al.'s approach both insert structures into the circuit without removing anything from the original circuit, if the attacker can find these structures, then they can be removed from the obfuscated circuit returning the original circuit. One proposal for thwarting such

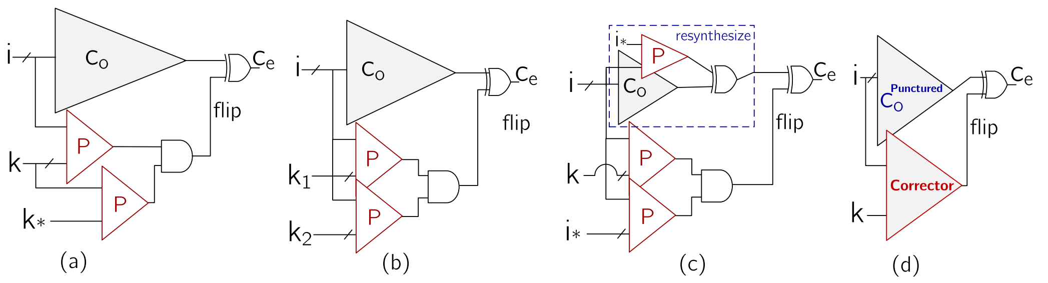


Fig. 17. Different generations of point-functions schemes. (a) Insert and correct with a correct-key comparator. (b) Anti-SAT style with a double key vector. (c) Corrupt-and-correct by inserting a point-function, resynthesizing and then using a point-function-based corrector. (c) A general puncture and correct approach, where c_o is first punctured on select input patterns and is then corrected with a correction block. The defender must hide the location of the punctured/corrected inputs at all costs.

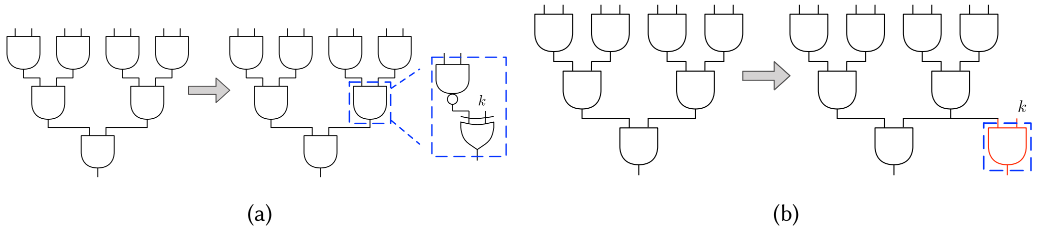


Fig. 18. Obfuscating the inserted AND-tree structure: (a) inserting key-logic in the tree and (b) connecting internal tree nodes to other circuit locations.

attacks was to obfuscate the point-function structure itself. This can be done by adding additional key-gates into the AND-trees as seen in Figure 18(a) or by connecting internal nodes to other circuit nodes through mutable key-logic (a key-gate that kills the propagation of a signal by configuring the key) as seen in Figure 18(b).

It is not clear exactly how effective AND-tree obfuscation is against removal attacks, especially since AppSAT's wire-disagreement analysis can spot the tip of low-error blocks well into the de-obfuscation process [80], at which point the AND-tree obfuscation key bits are likely resolved. Regardless, CamouPerturb [105] was the first to address the removal attack issue with the "corrupt-and-correct" approach. The idea in CamouPerturb was to first corrupt the original circuit by flipping its output on a small set of input patterns in a way that is difficult for an attacker to recover the flipped locations. Then the tree logic is inserted to compensate for the flipped patterns as seen in Figure 17(c). In this way, even if the attacker identifies and removes the correction circuitry, they are left with a circuit that is incorrect on a few unknown input patterns. This theme was later extended in References [70, 71, 108] where a look-up-table was used to correct the circuit on select input patterns. This makes the corrupt-and-correct scheme as seen in Figure 17(d) a general strategy for thwarting exact attacks. Several works have followed that attack corrupt-and-correct schemes that leak the punctured patterns through structure or functionality [83, 86, 102, 117]. Given these, it is possible to surmise that as long as (1) the number of input patterns on which c_o is punctured/corrupted is small, (2) the punctured circuit does not leak the location of the puncturing, and (3) the correction circuit itself does not leak the location of the puncturing, then the corrupt-and-correct scheme is provably secure in the exact-attack threat model.

Security under the exact attack model ensures that the attacker will not recover a perfectly functioning circuit, i.e., there will be locations in the input space for which the pirated circuit may

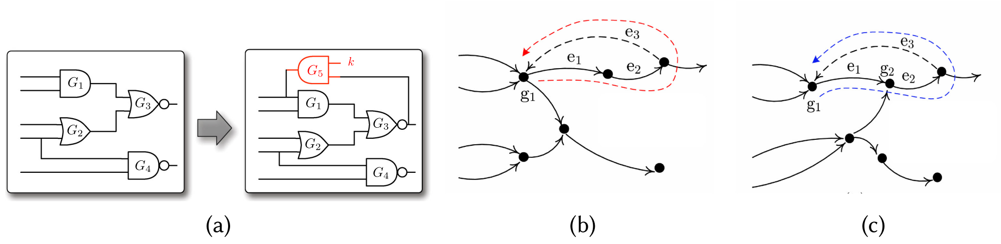


Fig. 19. (a) Example of cyclic obfuscation, (b) example of a reducible loop with a single entry point, and (c) an irreducible loop with two entry points (each node represents a gate and each edge represents a connection between two gates). The figures are adapted from Reference [77].

misbehave. This in itself can be useful as a **watermarking** technique as the defender can test these patterns on the circuit. Despite this, if the goal of the attacker is the easier goal of approximation, then point-function schemes are incredibly weak. In fact, the resilience of point-function schemes against exact attacks is exponentially inversely proportion to their approximation resiliency. For a p-bit point-function, 2^p queries are required to exactly recover the original functionality yet the error rate of the scheme is $1/2^p$. Compounding these schemes with traditional high-error schemes does not help due to attacks such as AppSAT and DDIP.

4.3.5 Cyclic Interconnect Obfuscation. Shamsi et al. was first to propose a cyclic obfuscation scheme [77]. Cyclic obfuscation is any obfuscation that creates an obfuscated circuit that is not a Directed-Acyclic-Graph (DAG) circuit. This can be achieved by inserting feedback paths controlled by key-bits. The idea is that even though the obfuscated circuit is cyclic, under the correct key the cycles open up and the circuit behaves combinationally. However, incorrect keys can lead to oscillating circuits. An example of cyclic obfuscation is shown in Figure 19(a).

Shamsi et al. contended that such cycles can thwart SAT attacks. When a cyclic circuit is converted to a SAT formula through the Tseitin transform, there are a few possibilities. If the circuit has internal oscillatory nodes, then the SAT formula representing the circuit is not satisfiable and the circuit is not combinational. Another possibility is that the circuit will have no oscillatory nodes, but the output of the circuit is not uniquely determined by its inputs. In this case there will be internal wires in the cyclic circuit, where a choice on their initial value will affect the function's output. Such a circuit is not combinational. If such a circuit is added to the mitter in the SAT attack, then the SAT attack will get stuck in an infinite loop, since the choice of internal wires will allow the SAT solver to keep satisfying the mitter regardless of the choice of keys and inputs just by flipping internal wires.

Shamsi et al. proposed two structural criteria to ensure that feedback paths cannot be removed by a simple graph traversal. The first criterion is that loops need to have multiple entry-points otherwise they are removable. For example, the loop in Figure 19(b) is reducible as it has a single entry point, i.e., g_1. Hence, by breaking the edge e_3, the loop can be safely removed. In contrast, the loop in Figure 19(c) is irreducible as it has two entry points, i.e., g_1 and g_2. The second criterion is that more than 2 edges in a loop have to be "removable" so that there exist multiple ways to break the loop. An edge is removable if the attacker is able to modify the key to remove an edge from the loop without creating gates with dangling inputs or outputs. For example in Figure 19(c), only e_3 can be removed, since if e_1 or e_2 is removed, then the output of g_1 or g_2 becomes dangling.

Zhou et al. [116] was the first to propose a SAT attack that is capable of breaking cyclically obfuscated circuits. This attack, called CycSAT, uses a pre-processing step to extract a "non-cyclic" condition on the key $NC(k)$. The $NC(k)$ condition is satisfied by keys that result in non-cyclic

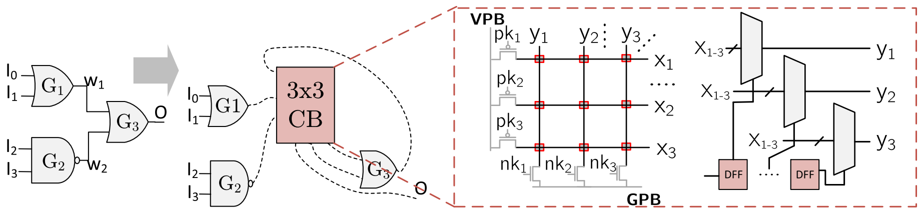


Fig. 20. Cyclic wire obfuscation leveraging a crossbar structure. The crossbar can be implemented with multiplexers and a scan-chain as well, however, with significantly more area overhead. Figures are adapted from Reference [78].

solution for c_e. By ANDing the mitter in the SAT attack with this condition the attack remains within the space of non-cyclic keys and terminates with a non-cyclic and correct solution. The condition is extracted by first finding a feedback-arc-set (FAS). A FAS is a set of edges that if opened up in a cyclic graph will make it acyclic. Each wire w in the FAS is opened up to w and w' where the transitive fanin on w' includes w. The NC condition is then constructed by ensuring that each w' is independent from w, i.e., the feedback arcs are opened. Using the CycSAT attack the authors were able to deobfuscate benchmark circuits obfuscated with Shamsi et al.'s cyclic obfuscation scheme [116] in runtime similar to SAT attacks on traditional schemes.

Rezaei et al. [65] and later Roshanisefat et al. [67] showed that a problem during the CycSAT attack is that when studying the cone that connects w to w' for extracting part of the NC condition, this cone is in fact cyclic itself. Analyzing the independence of w' and w when they are related by a cyclic circuit is a challenge. They contend in their paper that this challenge requires all simple cycles in the circuit to be enumerated and opened explicitly with non-cyclic conditions. Both papers propose cyclic obfuscation schemes that exploit this limitation in the CycSAT attack. In addition, if the original circuit itself is cyclic-but-combinational (such circuits exist and were first studied in Reference [6]), then the CycSAT attack becomes more complex, which in turn can be used to improve cyclic defenses, which we refer to as "cyclic+cyclic" schemes. In Reference [64], unreachable FSM states were used to implement cyclic+cyclic locking. Later, the Behavioral SAT attack (BeSAT) was proposed in Reference [82] to overcome CycSAT's limitation by detecting cyclic keys and banning them from the SAT solver on the fly as the attack advances.

Cyclic obfuscation schemes showcase the power of interconnect obfuscation as opposed to gate-insertion. From a silicon implementation perspective of interconnect locking, Shamsi et al. proposed an antifuse crossbar-based obfuscation that can create very dense cyclic interconnect ambiguity with little substrate overhead as the antifuse elements are inserted in the metal layers [78]. This scheme is shown in Figure 20, where a crossbar is used to program a two-dimensional array of antifuse devices bringing about area savings in terms of programming transistors. As for camouflaging, interconnect ambiguity can be created with much smaller overhead compared to gate-insertion using only dummy contacts and extra wires fitted into the empty routing spaces in the layout. Patnik et al. [55] proposed full-chip camouflaging by flooding the circuit with dummy contacts and wires. Their results on large circuits show significant resilience against exact and approximate SAT attacks with minimal overhead under the combinational oracle-guided model.

4.3.6 *Timing-based Parametric Obfuscation.*

Parametric obfuscation was proposed as a way to thwart oracle-guided attacks. The idea is to create an obfuscated circuit that is functionally equivalent to the original circuit and that the key only changes other properties of the circuit such as its timing. For instance, under an incorrect key the circuit will take much longer to process its inputs.

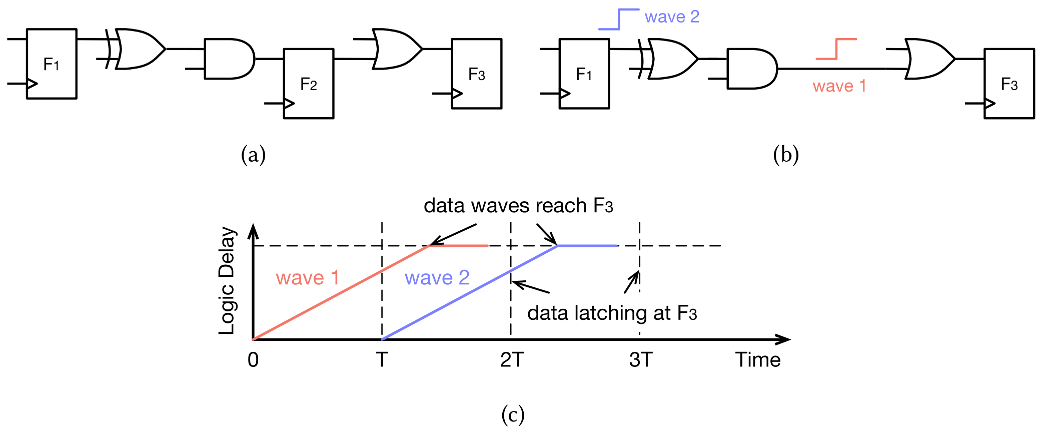


Fig. 21. Conventional timing and wave-pipelining: (a) Single-period clocking; (b) pipelining with two data waves; (c) timing/spatial diagram for wave propagation on a wave-pipelining path. The figures are adapted from Reference [114].

In Reference [100], besides regular XOR/XNOR-based obfuscation building blocks, gates that have delay values that are tunable by setting key-bits are inserted in the circuit. Without knowledge of the correct key, the attacker cannot operate the chip, since an incorrect key may result in violation of setup/hold timing constraints and corrupt outputs. Unfortunately, this scheme falls for the pitfall we discussed in Section 3, since it only prevents attackers from activating a pristine locked IC. The scheme in fact does not hide the Boolean function at all, since the tunable delay elements do not alter the Boolean expression. Furthermore, the attacker can simply set all delay elements to their highest possible delay and then slow down the clock and operate the circuit. Two works [5, 11] have followed this work with oracle-guided attacks that model the timing difference using equations on the key and use SAT-based and satisfiability-modulo-theory (SMT) solver-based algorithms to query and deobfuscate the circuit.

Given a parametric locking scheme that does not hide the Boolean functionality, all the attacker has to do to pirate the design is to resynthesize the recovered Boolean expression into a fast and timing-accurate logic. This is a trivial task for combinational logic given state-of-the-art synthesis and optimization tools, which is why the scheme discussed above [100] is insecure. For the case of larger sequential designs, this, however, can become challenging. Such an approach was presented in Reference [41] in which the FSM is augmented with stall cycles that activate if an incorrect key is used and hence an incorrect key can result in a much slower circuit. A somewhat different variant was later discussed in Reference [109]. These schemes have not received the same attack attention as in Reference [100]. Note that it is also possible to use point-functions to corrupt a critical signal in the design and hurt its performance as shown in Reference [113].

A timing-based locking scheme that avoids the flaw of Reference [100] is the one in Reference [114], which obfuscates the circuit timing by introducing multi-cycle paths. As shown in Figure 21(a), conventionally, all the paths in a combinational block operate within a single clock cycle. Reference [114] deliberately removes some flip-flops, e.g., F_2 in Figure 21(a), to convert single-cycle paths into wave-pipelining paths. On a wave-pipelining path, e.g., the combinational path from F_1 to F_3 in Figure 21(b), there are more than one data waves propagating without a flip-flop separating them. To retain the same functionalities as the original single-cycle circuit, the second wave cannot catch the first wave at any time during propagation as shown in Figure 21(c). Hence, the following timing constraints must be satisfied for all the wave-pipelining paths with two logic

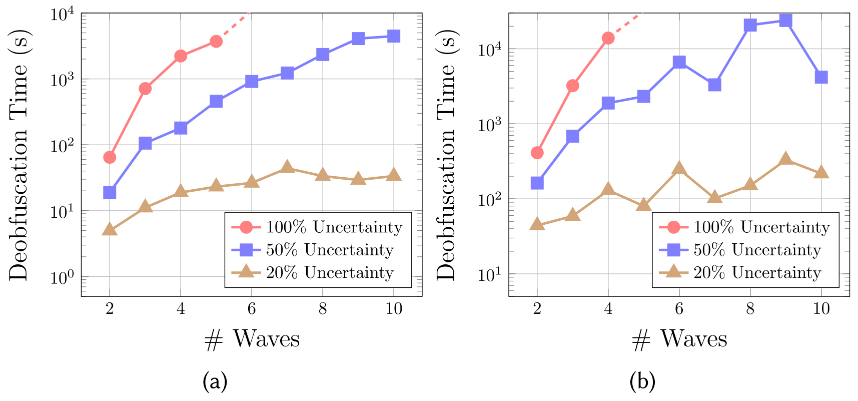


Fig. 22. Dependency of the AppSAT runtime on the timing uncertainty and the number of waves on the wave-pipelining paths for ISCAS'89 benchmarks (a) s5378 and (b) s13207. The figures are adapted from Reference [42].

waves:

$$T_{clk} + t_h \leq d_p \leq 2T_{clk} - t_{su}, \tag{1}$$

where d_p is the delay of the wave-pipelining paths and T_{clk} represents the clock period time. t_h and t_{su} denote the hold and setup time for a flip-flop.

By carefully removing the flip-flops and sizing the gates, the functionality of the design remains unchanged with a mixture of single-cycle and wave-pipelining paths created. Due to the existence of wave-pipelining paths, the attackers cannot recover the correct circuit functionality by simply changing the clock period time. Therefore, they are forced to determine the timing scheme for each path to decide whether it is single-cycle or wave-pipelining.

To determine the path timing, physical reverse engineering alone is insufficient as it is typically difficult to measure the gate and interconnect delay accurately [114]. Assuming the delay extraction techniques incur an inaccuracy factor τ $(0 < \tau < 1)$, for a path with delay d_p, if d_p satisfies

$$(1 - \tau)d_p \leq T_{clk} \leq (1 + \tau)d_p, \tag{2}$$

then the attackers cannot decide whether the path is wave-pipelined by physical reverse engineering.

Although the existing SAT attacks cannot be directly applied to such a parametric obfuscation scheme, Li et al. proposed a transformation procedure that can enable a SAT attack to work around this limitation [42]. This attacks was called TimingSAT. The security of this scheme depends heavily on two factors, the timing uncertainty τ, and the number of waves on the wave-pipelining paths. Figure 22 shows the runtime of the TimingSAT attack versus these two parameters.

4.3.7 Sequential Obfuscation. The term sequential obfuscation is used in literature for two very distinct concepts. First is a combinationally locked circuit where the attacker does not have access to some internal state-elements, which is modeled by the sequential-oracle-guided (SOG) attack model. Since attacks in the SOG model are quite novel, there has not yet been an extensive effort on defenses targeting this threat model. The second branch of research that is referred to as sequential obfuscation is that of FSM-obfuscation/transformation that we discuss herein.

An array of different research efforts on obfuscating FSMs can be broadly described as follows: The idea is to take an FSM from the original circuit, and add a number of additional dummy states to this FSM without in fact corrupting the original FSM. This way the original FSM will be embedded

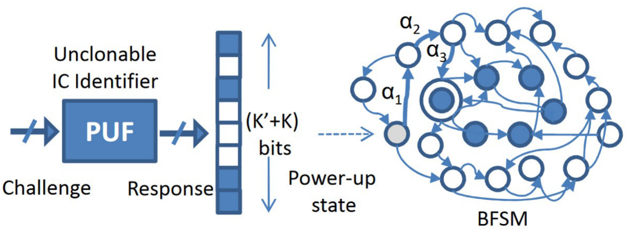


Fig. 23. Boosted FSM obfuscation. The original FSM (blue states) are augmented with many dummy states. The PUF decides the startup state, which is conveyed back to the owner, who can then in turn inform the end user of the correct path from the startup state to the original FSM.

in the sea of dummy states. The transitions are also constructed in a way to ensure that there exists paths from some of these dummy states to the original FSM. Some of these dummy states will have no path to the original FSM and arriving at one of them means there is no way to escape back to the original FSM and are hence referred to as black-hole states. With such an FSM, if one wants to operate the circuit with its original functionality, if the circuit starts up in one of the dummy states, then the dummy states need to be traversed to arrive at the original embedded FSM. Figure 23 demonstrates this augmented/boosted FSM.

There are several variations of the above idea. A prominent one referred to as Hardware Metering [37] uses FSM-obfuscation to create an IP protection protocol. This scheme, which is shown in Figure 23, begins by adding augmenting the original FSM with dummy and black-hole states. A unique random string extracted from a PUF is used to boot up the FSM in a random state. With high probability, this state will be one of the dummy states rather than the original FSM. Then the end-user sends this state to the owner of the IP. The owner, being the only party that has a map of the entire FSM, can send to the user the sequence of inputs that are necessary to traverse the dummy states to the original FSM. This creates a unique-per-chip unlocking protocol that allows the owner to keep track of each fabricated IC without disclosing the entire FSM (the master key) to the end-user.

Another FSM-obfuscation approach was initiated by HARPOON [13]. HARPOON worked by inserting an FSM with sequences of dummy states into a sequential or combinational circuit. This FSM is then used to feed the key wires of a combinational locking scheme. The FSM would output the correct key for the combinationally locked circuit only when it was traversed to the unlock state. This way a combinational circuit was turned into a sequential circuit that needs a set of secret input patterns before it can start operating correctly. Note that the FSM-based performance locking discussed earlier [41] can also be considered as an FSM-based obfuscation scheme.

Compared to combinational obfuscation schemes, the above sequential schemes have received little attention from an attack perspective. As a result, a well developed threat model for these schemes was never discussed in detail. Recently a couple of research efforts were directed at these schemes [28, 52] under an oracle-less threat model. Both attacks rely on FSM enumeration. These attacks are succesful due to the fact that one of the core assumptions in FSM-obfuscation is that the attacker cannot extract and enumerate the FSM or at least the part of the FSM that is important. However, even though a b-bit FSM has 2^b different state-vector values, not all of these states are reachable from the start-up state. In fact most FSM-obfuscation schemes generate the FSM by writing down the states in HDL and synthesizing it. This results in the fundamental contention: If the FSM is small enough to fit into HDL in a state-explicit form and be synthesized, then it must be small enough for an adversary that knows the state-registers to enumerate the states. Once the attacker has the graph of the FSM, due to the fixed style of the construction of these

FSMs it is typically possible to simply observe the unlocking path on the graph. [28] provides an extensive analysis of various FSM-obfuscation schemes.

Authors in Reference [28], after breaking various FSM-obfuscation schemes, propose a new method based on partial reconfiguration on an FPGA so that the circuit is loaded little by little as to avoid FSM enumeration attacks. However, it is not clear why an FPGA is used, since the goal of locking/camouflaging in the first place was that the defender does not want to pay the overhead price for using an FPGA. Otherwise an FPGA is a highly secure obfuscated circuit itself. An FPGA is a fully programmable circuit and hence the strongest form of logic locking possible. This is another pitfall in logic obfuscation research that we discuss shortly.

To summarize, unfortunately the proposed sequential obfuscation schemes so far do not seem to even withstand oracle-less attacks. It is expected that moving on to oracle-guided attacks will further weaken them. Despite this it is obvious that the richer semantics of sequential logic allow for much more complex obfuscated circuits and the development of IP protection protocols. This will in turn call for more sophisticated deobfuscation attacks as well.

4.3.8 Analog and Mixed-Signal Obfuscation. Analog circuits have received a lot less attention when it comes to obfuscation. Even though analog circuits are a smaller share of the overall electronics market, the IP value inherent to an analog design can surpass many digital blocks for which open-source RTL is becoming more and more available. Plus, modern analog designs are the result of thousands of person-hours of analysis and simulation. That said, unlike digital Boolean logic where the functionality of a circuit can be easily resynthesized into a different technology node, analog design is heavily tied to the particular technology node and it is difficult if not impossible to simply pirate a design from one technology node to a different one or even a different fab at the same node. This is due to the fact that properties important to analog design such as resistance, capacitance, and transistor trans-conductance vary vastly from node to node and fab to fab [4, 57].

Not surprisingly, the few works that have targeted protection of analog designs have focused on creating digitally programmable resistor/capacitors/current-sources and using them in the design [35, 63, 94]. This way a digital key will configure an analog property in the circuit. The benefit of these schemes is that not only do they provide security they can also improve reconfigurability to tune analog circuits post-fabrication to suppress process variation and improve reliability. Oracle-guided or oracle-less attacks in general on a locked analog circuit are possible by modelling the circuit as a real function rather than a Boolean one for which regression, machine-learning, non-linear optimization, and SMT-solving can be used to find key values that satisfy different criteria such as stability/gain for oracle-less attacks, and agreement-with-oracle for oracle-guided attacks.

One limitation with existing analog locking schemes is their focus on parameter hiding, which as discussed earlier is achieved automatically to some degree due to process variation and fab dependency especially for end-user attackers. It will be great moving forward to have schemes that can hide the high-level topology and architecture of analog circuit blocks as well at the cost of perhaps more overhead for the additional protection.

5 SUMMARY, PITFALLS, AND FUTURE DIRECTIONS

Figure 24 presents a map of the current state of obfuscation research with significant attack and defense ideas marked with blue and red squares, respectively. The attacks are marked with their attack model (OG, OL, and SOG). As can be seen from this map, locking/camouflaging schemes began with random insertion of key-structures (XOR/MUX/LUT) into the circuit. These schemes were broken with various generations of SAT attacks but only in the oracle-guided attack model. In the oracle-less model, attacks have been mildly successful in defeating some forms of XOR/XNOR insertion. Point-function schemes have succeeded in thwarting exact attacks. However, in the

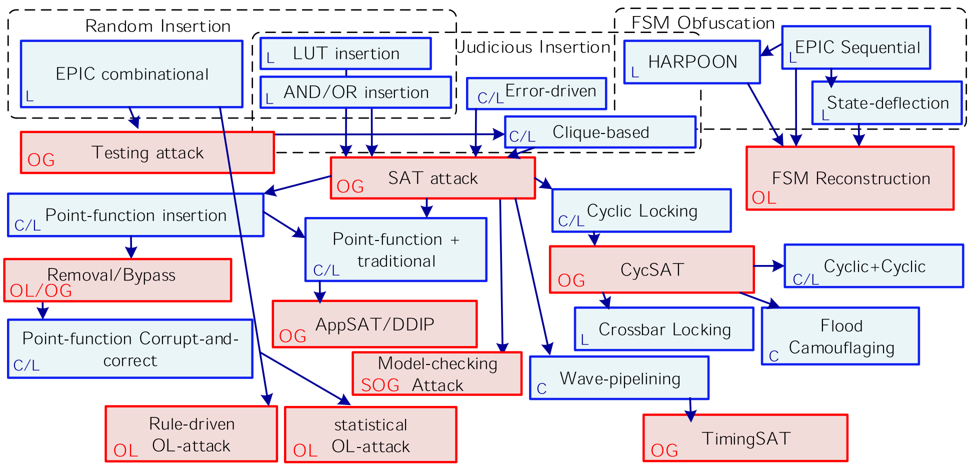


Fig. 24. Overview of the evolution of hardware obfuscation with oldest work at the top and newest work at the bottom. The red boxes denote attacks and blue boxes denote defenses. The OG (oracle-guided), SOG (sequential oracle-guided), and OL (oracle-less) tags describe the attack's threat model. The C (camouflaging) and L (locking) tags describe the category of defenses.

approximate attack model, there seems to be no straightforward approach for securing most circuits. Densely cyclic circuits may provide some hope for this path, since at this point nested cycles are difficult to attack with CycSAT attacks. Flood camouflaging in which ubiquitous ambiguity is created in the circuit with added interconnects and dummy wires stand undefeated, possibly settling the camouflaging problem. FSM-obfuscation schemes despite their attractiveness as a future direction are largely broken even in the weak oracle-less attack model.

We now list some important concise lessons and pitfalls that may prove useful to researchers and designers regarding the state of the art of locking and camouflaging:

- *The importance of the threat model*: at this point in obfuscation research we know the importance of the threat model. To put this in context, while we currently have working solutions with low overhead against oracle-less attacks and exact oracle-guided attacks, we are struggling to implement approximation resiliency with reasonable overhead and we know protection in the t-probed-oracle model is likely impossible. Hence it is important that designers decide their threat model before adopting an obfuscation scheme, which will determine what can be expected from the scheme in terms of security per overhead.
- *An FPGA/PLA fabric is the most secure form of locking and camouflaging*: The layout of a fully programmable fabric such as an FPGA/PLA provides little information about the circuit that is programmed into it. Unlike ASICs, with an FPGA the entire functionality of the circuit is encoded into electrical signals rather than physical features. However, an FPGA implementation of a design incurs significant overhead to the design, which is the motivation for locking/camouflaging ASICs. This also means that the frame of reference for overhead analysis is an FPGA implementation. If a locking/camouflaging scheme on a particular ASIC design results in overhead that is larger than implementing that design on an FPGA, then the scheme is not a viable low-overhead solution.
- *Active probing will render oracle-guided defenses useless*: If the attacker can probe internal wires of an oracle circuit with photon-emission, laser-probing, or active probing [47, 48], then securing locking and camouflaging becomes impossible. This means that for designers

that are adopting an oracle-guided attack model, defending against probing attacks should precede obfuscation. Preventing probing attacks itself is a hot area of research that may result in cheap and prevalent physical solutions in coming years.

- *Scan-chain security*: Preventing access to internal state-elements that do not need to be accessed outside of IC testing can improve the resiliency of locking/camouflaging against oracle-guided attacks by forcing the attacker to resolve to the more computationally complex sequential oracle-guided attacks. Considering the low cost incurred to the design for securing scan-chains this should be practiced in both locking and camouflaging. The strongest scan-chain protection scheme is simply deactivating the chain (zeroing the scan output) unless a "scan-unlock" key is received from the user, which can be checked with a comparator and a tamper-resistant secret string. Any attempt to obfuscate the scan output by mixing it with the scan-unlock key will leak key information unnecessarily.
- *IC camouflaging versus logic locking*: In IC camouflaging, since a programming circuitry does not need to be routed to all the ambiguous elements, the empty areas in a layout can be filled with dummy contacts, wires, and gates. This ubiquitous ambiguity that can be created in the case camouflaging significantly complicates even oracle-guided attacks on such dummy-filled layouts. Designers may use such camouflaging techniques to prevent end-user reverse engineering with high confidence, since research attacks at this point have not yet demonstrated successful reverse engineering of such schemes. These schemes also typically incur little overhead to the design. It is possible that such schemes can settle the camouflaging problem barring probing attacks.
- *Oracle-less locking*: Knowing how much more difficult it is to prevent oracle-guided attacks it may be a viable pathway for designers to ensure that adversaries do not get access to an oracle circuit. While for IC camouflaging this is almost impossible, for the case of locking this may be implemented. If designs are fabricated in batches where each batch is obfuscated with a different random seed, then designers can improve the probability of avoiding oracle-guided attacks.
- *Logic obfuscation and hardware Trojans*: Whether logic locking can prevent the insertion of hardware Trojans is heavily dependent on the particular design. Most hardware Trojans only need to attach to the ports of critical modules to complete an attack. Hence, if a module is locked yet the attacker is able to locate its ports, a Trojan insertion can be successful. Reference [46] shows a case study of locking and hardware Trojan resiliency on a RISC-V processor.
- *Functional secrecy versus chip inoperability*: Ensuring that a fabricated IC cannot be activated is an inherently different task than ensuring that a reverse engineering attacker cannot learn the functionality of the circuit. The goal of locking and camouflaging is the second task, which, if implemented securely, will achieve the first task automatically. An immediate result of this proposition is that in locking schemes any operation that operates strictly on the key without combining with the input (i.e., wires that only depend on key inputs and not primary inputs) does not contribute to functional secrecy and can be removed from the circuit by an attacker. Locking/camouflaging is about intimately mixing ambiguity with a circuit's functionality and structure in a non-removable way.
- *Functional secrecy versus indistinguishability obfuscation*: Indistinguishability Obfuscation or *iO* is a hot research topic in theoretical cryptography [29]. *iO* is an operator that can convert a circuit c_1 to a circuit $iO(c_1)$ where $iO(c_1)$ is statistically indistinguishable from $iO(c_2)$ where c_2 is any circuit that implements the same functionality as c_1. In other words, what *iO* does is destroy the "uniqueness" of c_1 so that it resembles any other circuit that implements the same functionality. For instance the Binary-Decision-Diagram (BDD) of

any circuit is a perfect indistinguishability obfuscation of that circuit, since a BDD representation is canonical, meaning that it is the same for all circuits that implement the same function. *iO* is fundamentally different than locking, since in *iO* the functionality is revealed to the attacker, it is just that the unique way that the design implements that functionality is what is hidden by *iO*. Locking requires the extra step of obscuring the functionality given a set of secret key variables and given the ability to hide from the attacker internal wires, which is not available to iO [41, 115]. While for small combinational circuits iO will not satisfy the goals of locking, for larger sequential/algorithmic hardware, iO and structural transformations may be used to hide the unique implementation of a circuit/program [41, 72].

Some open challenges in the area are as follows:

- *Physical Aspects*: Active reading of electrical signals on a chip is a serious threat against all sorts of hardware security measures including but not limited to obfuscation. Tamper-resistant nano-devices and architectures tie with this problem of physical security plus the huge effect they have on the security-per-overhead of locking/camouflaging.
- *Approximation-resiliency*: We do not yet have a clear idea on how to reason about approximation resiliency for combinational circuits. Most small combinational circuits seem to not withstand approximate attacks regardless of the traditional scheme that is used with low overhead. It may be that achieving such a property for smaller circuits is simply impossible but a formal/mathematical demonstration of this is an open challenge.
- *Cyclic attacks*: While we have various cyclic obfuscation scheme in the area of cyclic attacks, it seems that the CycSAT attacks may have limitations. Designing faster CycSAT attacks will help understand if the resiliency of cyclically obfuscated circuits is a genuine computational hardness or is due to simply not having the right algorithms in place at this time.
- *Formal proofs of security*: Modern theoretical cryptography relies heavily on formalism for reasoning about security. Security properties are clearly defined in mathematical terms and reduced to the hardness of other problems. Successfully applying this to logic obfuscation is an open challenge. Note that many faster cryptographic primitives such as block-ciphers rely on heuristic security, yet there is a rich mathematical framework for reasoning about such primitives that is simply not present in locking/camouflaging even though the two domains are quite related.
- *Sequential obfuscation/deobfuscation*: while the majority of current research targets combinational circuits, a move to reasoning about obfuscated sequential circuits is much needed. Better modelling of threats and goals of obfuscation in dealing with sequential designs will help avoid some of the mistakes of past schemes. Sequential obfuscation/deobfuscation will directly translate to reasoning about obfuscation at higher levels of the design such as RTL level and SoC level or even the software-hardware interface.
- *New notions of security*: In this article, we focused on functional secrecy as the security property. However, defining novel notions of security that call for different approaches are an attractive research goal. *iO* and parametric obfuscation are examples of properties that are different than functional secrecy yet may be made useful in specific scenarios.

6 CONCLUSION

The globalization of the IC supply chain propels the growth of the semiconductor industry and makes it the backbone of the modern-day computing system. However, it comes at the cost of hardware IP privacy risks due to reverse engineering. Consequently, the research on hardware IP protection through logic obfuscation has grown over the past decade and received considerable attention. This article provides a detailed survey of the recent progress in this area. The article first laid down a foundation for reasoning about security of these schemes by exploring various threat models. It then surveyed the state of the art in deobfuscation attacks within each attack model including the testing-based attack, the SAT attack and several of its variants and oracle-less attacks. The article then surveyed state of the art in defenses by dissecting locking/camouflaging into different layers of abstraction including the cell level, and then netlist level. We discussed representative netlist-level schemes and then surveyed representative attacks. Despite the existing research, logic obfuscation remains an important area with new challenges to be resolved, including novel attack models, understanding higher-level notions of obfuscation, and formal proofs of security. These challenges will certainly attract more research into this domain in the future.

REFERENCES

[1] Praneet Adusumilli, Alexander Reznicek, Oscar van der Straten, Miaomiao Wang, and Chih-Chao Yang. 2018. Metal finfet anti-fuse. US Patent App. 15/968,235.

[2] Nail Etkin Can Akkaya, Burak Erbagci, and Ken Mai. 2018. A secure camouflaged logic family using post-manufacturing programming with a 3.6 GHz adder prototype in 65nm CMOS at 1V nominal V DD. In *Proceedings of the 2018 IEEE International Solid-State Circuits Conference (ISSCC'18).* IEEE, 128–130.

[3] Yousra Alkabani and Farinaz Koushanfar. 2007. Active hardware metering for intellectual property protection and security. In *Proceedings of the USENIX Conference on Security.* 291–306.

[4] Angelos Antonopoulos, Christiana Kapatsori, and Yiorgos Makris. 2017. Security and trust in the analog/mixed-signal/RF domain: A survey and a perspective. In *Proceedings of the 2017 22nd IEEE European Test Symposium (ETS'17).* IEEE, 1–10.

[5] Kimia Zamiri Azar, Hadi Mardani Kamali, Houman Homayoun, and Avesta Sasan. 2019. SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), 97–122.

[6] John Backes, Brian Fett, and Marc D. Riedel. 2008. The analysis of cyclic circuits with Boolean satisfiability. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design.* IEEE Press, 143–148.

[7] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. 2010. Preventing IC piracy using reconfigurable logic barriers. *IEEE Des. Test Comput.* 27, 1 (2010).

[8] Alex Clark Baumgarten. 2009. Preventing integrated circuit piracy using reconfigurable logic barriers. In *IEEE Design and Test of Computers* 27, 1 (2010), 66–75. DOI: 10.1109/MDT.2010.24

[9] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. 2014. Stealthy dopant-level hardware Trojans: Extended version. *J. Cryptogr. Eng.* 4, 1 (2014), 19–31.

[10] J. al Birkner, A. Chan, H. T. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze, and R. Wong. 1992. A very-high-speed field-programmable gate array using metal-to-metal antifuse programmable elements. *Microelectr. J.* 23, 7 (1992), 561–568.

[11] Abhishek Chakraborty, Yuntao Liu, and Ankur Srivastava. 2018. TimingSAT: Timing profile embedded SAT attack. In *Proceedings of the International Conference on Computer-Aided Design.* ACM, 6.

[12] Prabuddha Chakraborty, Jonathan Cruz, and Swarup Bhunia. 2018. SAIL: Machine learning guided structural analysis attack on hardware obfuscation. In *Proceedings of the Asian Hardware Oriented Security and Trust Symposium (AsianHOST'18).* 56–61.

[13] R. S. Chakraborty and S. Bhunia. 2009. HARPOON: An obfuscation-based SoC design methodology for hardware protection. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 28, 10 (2009), 1493–1502.

[14] Rajat Subhra Chakraborty and Swarup Bhunia. 2009. Security against hardware Trojan through a novel application of design obfuscation. In *Proceedings of the International Conference on Computer Aided Design.* 113–116.

[15] Shuai Chen, Junlin Chen, Domenic Forte, Jia Di, Mark Tehranipoor, and Lei Wang. 2015. Chip-level anti-reverse engineering using transformable interconnects. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems.* 109–114.

[16] Chipworks. 2012. Intel's 22-nm Tri-gate Transistors Exposed. Retrieved from http://www.eet.bme.hu/ mizsei/Montech/intel-s-22-nm-trigate-transistors-exposed.html.

[17] Lap-Wai Chow, James P. Baukus, and William M. Clark Jr. 2007. Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide. US Patent 7,294,935.

[18] Lap-Wai Chow, William M. Clark Jr, and James P. Baukus. 2007. Covert transformation of transistor properties as a circuit protection method. US Patent 7,217,977.

[19] Ronald P. Cocchi, James P. Baukus, Lap Wai Chow, and Bryan J. Wang. 2014. Circuit camouflage integration for hardware IP protection. In *Proceedings of the IEEE/ACM Design Automation Conference.* Article 153, 5 pages.

[20] Ronald P. Cocchi, James P. Baukus, Bryan J. Wang, Lap Wai Chow, and Paul Ouyang. 2012. Building block for a secure CMOS logic cell library. US Patent 8,111,089.

[21] Jason Cong and Bingjun Xiao. 2014. FPGA-RPI: A novel FPGA architecture with RRAM-based programmable interconnects. *IEEE Trans. VLSI Syst.* 22, 4 (2014), 864–877.

[22] Actel Corp. 2002. Design Security in Nonvolatile Flash and Antifuse FPGAs. Actel Corp. Technical Report on Quick-Logic FPGAs. Retrieved from http://www.actel.com/documents/DesignSecurity_WP.pdf.

[23] Franck Courbon, Sergei Skorobogatov, and Christopher Woods. 2016. Reverse engineering flash EEPROM memories using scanning electron microscopy. In *Proceedings of the International Conference on Smart Card Research and Advanced Applications*. Springer, 57–72.

[24] Niklas Eén and Niklas Sörensson. 2003. An extensible SAT-solver. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*. Springer, 502–518.

[25] Mohamed El Massad, Siddharth Garg, and Mahesh Tripunitara. 2017. Reverse engineering camouflaged sequential circuits without scan access. In *Proceedings of the International Conference on Computer Aided Design*. IEEE, 33–40.

[26] Mohamed El Massad, Siddharth Garg, and Mahesh V. Tripunitara. 2015. Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes. In *Proceedings of the Network and Distributed System Security Symposium.*

[27] Burak Erbagci, Cagri Erbagci, Nail Etkin Can Akkaya, and Ken Mai. 2016. A secure camouflaged threshold voltage defined logic family. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust.* 229–235.

[28] Marc Fyrbiak, Sebastian Wallat, Jonathan Déchelotte, Nils Albartus, Sinan Böcker, Russell Tessier, and Christof Paar. 2018. On the difficulty of FSM-based hardware obfuscation. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 293–330.

[29] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. 2016. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.* 45, 3 (2016), 882–929.

[30] Adria Gascón, Pramod Subramanyan, Bruno Dutertre, Ashish Tiwari, Dejan Jovanović, and Sharad Malik. 2014. Template-based circuit understanding. In *Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design.* FMCAD Inc, 83–90.

[31] Jonathan Greene, Sinan Kaptanoglu, Wenyi Feng, Volker Hecht, Joel Landry, Fei Li, Anton Krouglyanskiy, Mihai Morosan, and Val Pevzner. 2011. A 65nm flash-based FPGA fabric optimized for low cost and power. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, 87–96.

[32] Ujjwal Guin, Ziqi Zhou, and Adit Singh. 2017. A novel design-for-security (DFS) architecture to prevent unauthorized IC overproduction. In *Proceedings of the 2017 IEEE 35th VLSI Test Symposium (VTS'17).* IEEE, 1–6.

[33] Ujjwal Guin, Ziqi Zhou, and Adit Singh. 2018. Robust design-for-security architecture for enabling trust in ic manufacturing and test. *IEEE Trans. VLSI Syst.* 26, 5 (2018), 818–830.

[34] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh Tripunitara. 2013. Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation. In *Proceedings of the USENIX Conference on Security*. USENIX, 495–510.

[35] Nithyashankari Gummidipoondi Jayasankaran, Adriana Sanabria Borbon, Edgar Sanchez-Sinencio, Jiang Hu, and Jeyavijayan Rajendran. 2018. Towards provably-secure analog and mixed-signal locking against overproduction. In *Proceedings of the International Conference on Computer-Aided Design*. ACM, 7.

[36] Takashi Kono, Tomoya Saito, and Tadaaki Yamauchi. 2018. Overview of embedded flash memory technology. In *Embedded Flash Memory for Embedded Systems: Technology, Design for Sub-systems, and Innovations*. Springer, 29–74.

[37] Farinaz Koushanfar. 2012. Hardware metering: A survey. In *Introduction to Hardware Security and Trust.* Springer, 103–122.

[38] Ian Kuon, Russell Tessier, Jonathan Rose, et al. 2008. FPGA architecture: Survey and challenges. *Found. Trends Electr. Des. Autom.* 2, 2 (2008), 135–253.

[39] Y. W. Lee and N. A. Touba. 2015. Improving logic obfuscation via logic cone analysis. In *Proceedings of the IEEE Latin-American Test Symposium.*1–6.

[40] Leon Li and Alex Orailoglu. 2019. Piercing logic locking keys through redundancy identification. In *Design, Automation and Test in Europe Conference & Exhibition (DATE'19).* 540–545. DOI: 10.23919/DATE.2019.8714955

[41] Li Li and Hai Zhou. 2013. Structural transformation for best-possible obfuscation of sequential circuits. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust.* 55–60.

[42] Meng Li, Kaveh Shamsi, Yier Jin, and David Z. Pan. 2018. TimingSAT: Decamouflaging timing-based logic obfuscation (unpublished).

[43] Meng Li, Kaveh Shamsi, Travis Meade, Zheng Zhao, Bei Yu, Yier Jin, and David Z. Pan. 2016. Provably secure camouflaging strategy for IC protection. In *Proceedings of the International Conference on Computer Aided Design*. Article 28, 8 pages.

[44] Meng Li, Bei Yu, Yibo Lin, Xiaoqing Xu, Wuxi Li, and David Z. Pan. 2018. A practical split manufacturing framework for Trojan prevention via simultaneous wire lifting and cell insertion. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 265–270.

[45] Wenchao Li, A. Gascon, P. Subramanyan, Wei Yang Tan, A. Tiwari, S. Malik, N. Shankar, and S. A. Seshia. 2013. WordRev: Finding word-level structures in a sea of bit-level gates. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*. 67–74.

[46] Timothy Linscott, Pete Ehrett, Valeria Bertacco, and Todd Austin. 2018. SWAN: Mitigating hardware trojans with design ambiguity. In *Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'18)*. IEEE, 1–7.

[47] Heiko Lohrke, Shahin Tajik, Christian Boit, and Jean-Pierre Seifert. 2016. No place to hide: Contactless probing of secret data on FPGAs. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 147–167.

[48] Heiko Lohrke, Shahin Tajik, Thilo Krachenfels, Christian Boit, and Jean-Pierre Seifert. 2018. Key extraction using thermal laser stimulation. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 3 (Aug. 2018), 573–595. DOI: 10.13154/tches.v2018.i3.573-595

[49] S. Malik, G. T. Becker, C. Paar, and W. P. Burleson. 2015. Development of a layout-level hardware obfuscation tool. In *Proceedings of the IEEE Annual Symposium on VLSI*. 204–209.

[50] Mohamed El Massad, Siddharth Garg, and Mahesh Tripunitara. 2017. Reverse engineering camouflaged sequential integrated circuits without scan access. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD'17)*. 33–40. DOI: 10.1109/ICCAD.2017.8203757

[51] Mohamed El Massad, Jun Zhang, Siddharth Garg, and Mahesh V. Tripunitara. 2017. Logic locking for secure outsourced chip fabrication: A new attack and provably secure defense mechanism. *arXiv preprint arXiv:1703.10187* (2017).

[52] Travis Meade, Zheng Zhao, Shaojie Zhang, David Pan, and Yier Jin. 2017. Revisit sequential logic obfuscation: Attacks and defenses. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. IEEE, 1–4.

[53] Travis Meade, Zheng Zhao, Shaojie Zhang, David Z. Pan, and Yier Jin. 2017. Revisit sequential logic obfuscation: Attacks and defenses. In *Proceedings of the IEEE International Symposium on Circuits and Systems*.

[54] Arlindo L. Oliveira. 2001. Techniques for the creation of digital watermarks in sequential circuit designs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 20, 9 (2001), 1101–1117.

[55] Satwik Patnaik, Mohammed Ashra, Johann Knechtel, and Ozgur Sinanoglu. 2017. Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging. In *Proceedings of the International Conference on Computer Aided Design*. IEEE, 41–48.

[56] Stephen M. Plaza and Igor L. Markov. 2015. Solving the third-shift problem in IC piracy with test-aware logic locking. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 34, 6 (2015), 961–971.

[57] Ilia Polian. 2016. Security aspects of analog and mixed-signal circuits. In *Proceedings of the 2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW'16)*. IEEE, 1–6.

[58] Shahed E. Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. 2016. A survey on chip to system reverse engineering. *ACM J. Emerg. Technol. Comput. Syst.* 13, 1 (2016), 6:1–6:34.

[59] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. 2012. Security analysis of logic obfuscation. In *Proceedings of the IEEE/ACM Design Automation Conference*. 83–89.

[60] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. 2013. Security analysis of integrated circuit camouflaging. In *Proceedings of the ACM Conference on Computer & Communications Security*. 709–720.

[61] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. 2013. Security analysis of integrated circuit camouflaging. In *Proceedings of the ACM Conference on Computer & Communications Security*. 709–720.

[62] Jeyavijayan Rajendran, Huan Zhang, Chi Zhang, Garrett S. Rose, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. 2015. Fault analysis-based logic encryption. 64, 2 (2015), 410–424. DOI: 10.1109/TC.2013.193

[63] Vaibhav Venugopal Rao and Ioannis Savidis. 2017. Protecting analog circuits with parameter biasing obfuscation. In *Proceedings of the 2017 18th IEEE Latin American Test Symposium (LATS'17)*. IEEE, 1–6.

[64] Amin Rezaei, You Li, Yuanqi Shen, Shuyu Kong, and Hai Zhou. 2019. CycSAT-unresolvable cyclic logic encryption using unreachable states. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 358–363.

[65] Amin Rezaei, Yuanqi Shen, Shuyu Kong, Jie Gu, and Hai Zhou. 2018. Cyclic locking and memristor-based obfuscation against CycSAT and inside foundry attacks. In *Proceedings of the Design, Automation and Test in Europe*. IEEE, 85–90.

[66] Shervin Roshanisefat, Hadi Mardani Kamali, and Avesta Sasan. 2018. SRCLock: SAT-resistant cyclic logic locking for protecting the hardware. *arXiv preprint arXiv:1804.09162* (2018).

[67] Shervin Roshanisefat, Hadi Mardani Kamali, and Avesta Sasan. 2018. SRCLock: SAT-resistant cyclic logic locking for protecting the hardware. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*. ACM, 153–158.

[68] Deepu Roy, Johan H. Klootwijk, Nynke A. M. Verhaegh, Harold H. A. J. Roosen, and Rob A. M. Wolters. 2009. Comb capacitor structures for on-chip physical uncloneable function. *IEEE Trans. Semicond. Manufact.* 22, 1 (2009), 96–102.

[69] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. EPIC: Ending piracy of integrated circuits. In *Proceedings of the Conference on Design, Automation and Test in Europe*. 1069–1074.

[70] Abhrajit Sengupta, Mohammed Ashraf, Mohammed Nabeel, and Ozgur Sinanoglu. 2018. Customized locking of IP blocks on a multi-million-gate SoC. In *Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'18)*. IEEE, 1–7.

[71] Abhrajit Sengupta, Mohammed Nabeel, Muhammad Yasin, and Ozgur Sinanoglu. 2018. ATPG-based cost-effective, secure logic locking. In *Proceedings of the 2018 IEEE 36th VLSI Test Symposium (VTS'18)*. IEEE, 1–6.

[72] Anirban Sengupta, Dipanjan Roy, Saraju P. Mohanty, and Peter Corcoran. 2017. DSP design protection in CE through algorithmic transformation based structural obfuscation. *IEEE Trans. Consum. Electr.* 63, 4 (2017), 467–476.

[73] Bicky Shakya, Haoting Shen, Mark Tehranipoor, and Domenic Forte. 2019. Covert gates: Protecting integrated circuits with undetectable camouflaging. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 3 (May 2019), 86–118. DOI: https://doi.org/10.13154/tches.v2019.i3.86-118

[74] Bicky Shakya, Mark M. Tehranipoor, Swarup Bhunia, and Domenic Forte. 2017. Introduction to hardware obfuscation: Motivation, methods and evaluation. In *Hardware Protection through Obfuscation*. Springer, 3–32.

[75] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*.

[76] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. 2017. Circuit obfuscation and oracle-guided attacks: Who can prevail? In *Proceedings of the IEEE Great Lakes Symposium on VLSI*.

[77] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. 2017. Cyclic obfuscation for creating SAT-unresolvable circuits. In *Proceedings of the IEEE Great Lakes Symposium on VLSI*.

[78] Kaveh Shamsi, Meng Li, David Z. Pan, and Yier Jin. 2018. Cross-lock: Dense layout-level interconnect locking using cross-bar architectures. In *Proceedings of the IEEE Great Lakes Symposium on VLSI*.

[79] Kaveh Shamsi, Meng Li, David Z. Pan, and Yier Jin. 2019. KC2: Key-condition crunching for fast sequential circuit deobfuscation. In *Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE'19)*. IEEE, 534–539.

[80] Kaveh Shamsi, Travis Meade, Meng Li, David Z. Pan, and Yier Jin. 2019. On the approximation resiliency of logic locking and IC camouflaging schemes. *IEEE Trans. Inf. Forens. Secur.* 14, 2 (2019), 347–359.

[81] Yuanqi Shen, You Li, Shuyu Kong, Amin Rezaei, and Hai Zhou. 2019. SigAttack: New High-level SAT-based Attack on Logic Encryptions. Cryptology ePrint Archive, Report 2019/061. Retrieved from https://eprint.iacr.org/2019/061.

[82] Yuanqi Shen, You Li, Amin Rezaei, Shuyu Kong, David Dlott, and Hai Zhou. 2019. BeSAT: Behavioral SAT-based attack on cyclic logic encryption. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 657–662.

[83] Yuanqi Shen, Amin Rezaei, and Hai Zhou. 2018. SAT-based bit-flipping attack on logic encryptions. In *Proceedings of the Conference on Design, Automation and Test in Europe*. IEEE, 629–632.

[84] Yuanqi Shen and Hai Zhou. 2017. Double DIP: Re-evaluating security of logic encryption algorithms. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*. ACM, 179–184.

[85] Mitsuru Shiozaki, Ryohei Hori, and Takeshi Fujino. 2014. Diffusion programmable device: The device to prevent reverse engineering. *IACR Cryptology ePrint Archive* 2014 (2014), 109.

[86] Deepak Sirone and Pramod Subramanyan. 2018. Functional analysis attacks on logic locking. *arXiv preprint arXiv:1811.12088* (2018).

[87] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*. 137–143.

[88] P. Subramanyan, N. Tsiskaridze, Wenchao Li, A. Gascon, Wei Yang Tan, A. Tiwari, N. Shankar, S. A. Seshia, and S. Malik. 2014. Reverse engineering digital circuits using structural and functional analyses. *IEEE Trans. Emerg. Top. Comput.* 2, 1 (2014), 63–80.

[89] Takeshi Sugawara, Daisuke Suzuki, Ryoichi Fujii, Shigeaki Tawa, Ryohei Hori, Mitsuru Shiozaki, and Takeshi Fujino. 2014. Reversing stealthy dopant-level circuits. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 112–126.

[90] Shahin Tajik, Heiko Lohrke, Jean-Pierre Seifert, and Christian Boit. 2017. On the power of optical contactless probing: Attacking bitstream encryption of FPGAs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1661–1674.

[91] Randy Torrance and Dick James. 2009. The state-of-the-art in IC reverse engineering. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems.* Springer, 363–381.

[92] Pim Tuyls, Geert-Jan Schrijen, Boris Škorić, Jan Van Geloven, Nynke Verhaegh, and Rob Wolters. 2006. Read-proof hardware from protective coatings. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems.* Springer, 369–383.

[93] Arunkumar Vijayakumar, Vinay C. Patil, Daniel E. Holcomb, Christof Paar, and Sandip Kundu. 2017. Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques. *IEEE Trans. Inf. Forens. Secur.* 12, 1 (2017), 64–77.

[94] Jiafan Wang, Congyin Shi, Adriana Sanabria-Borbon, Edgar Sánchez-Sinencio, and Jiang Hu. 2017. Thwarting analog IC piracy via combinational locking. In *Proceedings of the 2017 IEEE International Test Conference (ITC'17).* IEEE, 1–10.

[95] Xueyan Wang, Xiaotao Jia, Qiang Zhou, Yici Cai, Jianlei Yang, Mingze Gao, and Gang Qu. 2016. Secure and low-overhead circuit obfuscation technique with multiplexers. In *Proceedings of the IEEE Great Lakes Symposium on VLSI.* ACM, 133–136.

[96] Xueyan Wang, Qiang Zhou, Yici Cai, and Gang Qu. 2016. Is the secure IC camouflaging really secure? In *Proceedings of the IEEE International Symposium on Circuits and Systems.* IEEE, 1710–1713.

[97] Xueyan Wang, Qiang Zhou, Yici Cai, and Gang Qu. 2018. A conflict-free approach for parallelizing SAT-based de-camouflaging attacks. In *Proceedings of the Asia and South Pacific Design Automation Conference.* 259–264.

[98] Xueyan Wang, Qiang Zhou, Yici Cai, and Gang Qu. 2018. Towards a formal and quantitative evaluation framework for circuit obfuscation methods. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* (2018).

[99] Yang Xie and Ankur Srivastava. 2016. Mitigating SAT attack on logic locking. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems.* 127–146.

[100] Yang Xie and Ankur Srivastava. 2017. Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction. In *Proceedings of the IEEE/ACM Design Automation Conference.*

[101] Xiaolin Xu, Bicky Shakya, Mark M. Tehranipoor, and Domenic Forte. 2017. Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems.* Springer, 189–210.

[102] Fangfei Yang, Ming Tang, and Ozgur Sinanoglu. 2019. Stripped functionality logic locking with hamming distance based restore unit (SFLL-hd)–Unlocked. *IEEE Transactions on Information Forensics and Security* (2019).

[103] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan Rajendran, and Ozgur Sinanoglu. 2016. SARLock: SAT attack resistant logic locking. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust.* 236–241.

[104] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan J. V. Rajendran, and Ozgur Sinanoglu. 2017. TTLock: Tenacious and traceless logic locking. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust.* IEEE, 166–166.

[105] Muhammad Yasin, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. 2016. CamoPerturb: Secure IC camouflaging for minterm protection. In *Proceedings of the International Conference on Computer Aided Design.* Article 29, 8 pages.

[106] Muhammad Yasin, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. 2017. Removal attacks on logic locking and camouflaging techniques. *IEEE Trans. on Information Forensics and Security* (2017).

[107] Muhammad Yasin, Samah Mohamed Saeed, Jeyavijayan Rajendran, and Ozgur Sinanoglu. 2016. Activation of logic encrypted chips: Pre-test or post-test? In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe.* EDA Consortium, 139–144.

[108] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan J. V. Rajendran, and Ozgur Sinanoglu. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of the ACM Conference on Computer & Communications Security.* ACM, 1601–1618.

[109] Muhammad Yasin, Abhrajit Sengupta, Benjamin Carrion Schafer, Yiorgos Makris, Ozgur Sinanoglu, and Jeyavijayan J. V. Rajendran. 2017. What to lock?: Functional and parametric locking. In *Proceedings of the IEEE Great Lakes Symposium on VLSI.* ACM, 351–356.

[110] Muhammad Yasin and Ozgur Sinanoglu. 2015. Transforming between logic locking and IC camouflaging. In *Proceedings of the 2015 10th International Design & Test Symposium (IDT'15).* IEEE, 1–4.

[111] Cunxi Yu, Xiangyu Zhang, Duo Liu, Maciej Ciesielski, and Daniel Holcomb. 2017. Incremental SAT-based reverse engineering of camouflaged logic circuits. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* PP, 99 (2017), 1–1.

[112] Lin Yuan, Gang Qu, T. Villa, and A. Sangiovanni-Vincentelli. 2008. An FSM reengineering approach to sequential circuit synthesis by state splitting. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 27, 6 (2008), 1159–1164.

[113] Monir Zaman, Abhrajit Sengupta, Danqing Liu, Ozgur Sinanoglu, Yiorgos Makris, and Jeyavijayan Rajendran. 2018. Towards provably-secure performance locking. In *Proceedings of the Conference on Design, Automation and Test in Europe.* 1592–1597.

[114] Li Zhang, Bing Li, Bei Yu, David Z. Pan, and Ulf Schlichtmann. 2018. TimingCamouflage: Improving circuit security against counterfeiting by unconventional timing. In *Proceedings of the Conference on Design, Automation and Test in Europe*.

[115] Hai Zhou. 2017. A humble theory and application for logic encryption. *IACR Cryptology ePrint Archive* 2017 (2017), 696.

[116] Hai Zhou, Ruifeng Jiang, and Shuyu Kong. 2017. CycSAT: SAT-based attack on cyclic logic encryptions. In *Proceedings of the International Conference on Computer Aided Design*. IEEE, 49–56.

[117] Hai Zhou, Yuanqi Shen, and Amin Rezaei. 2019. Vulnerability and Remedy of Stripped Function Logic Locking. Cryptology ePrint Archive, Report 2019/139. Retrieved from https://eprint.iacr.org/2019/139.

Received January 2019; revised June 2019; accepted June 2019